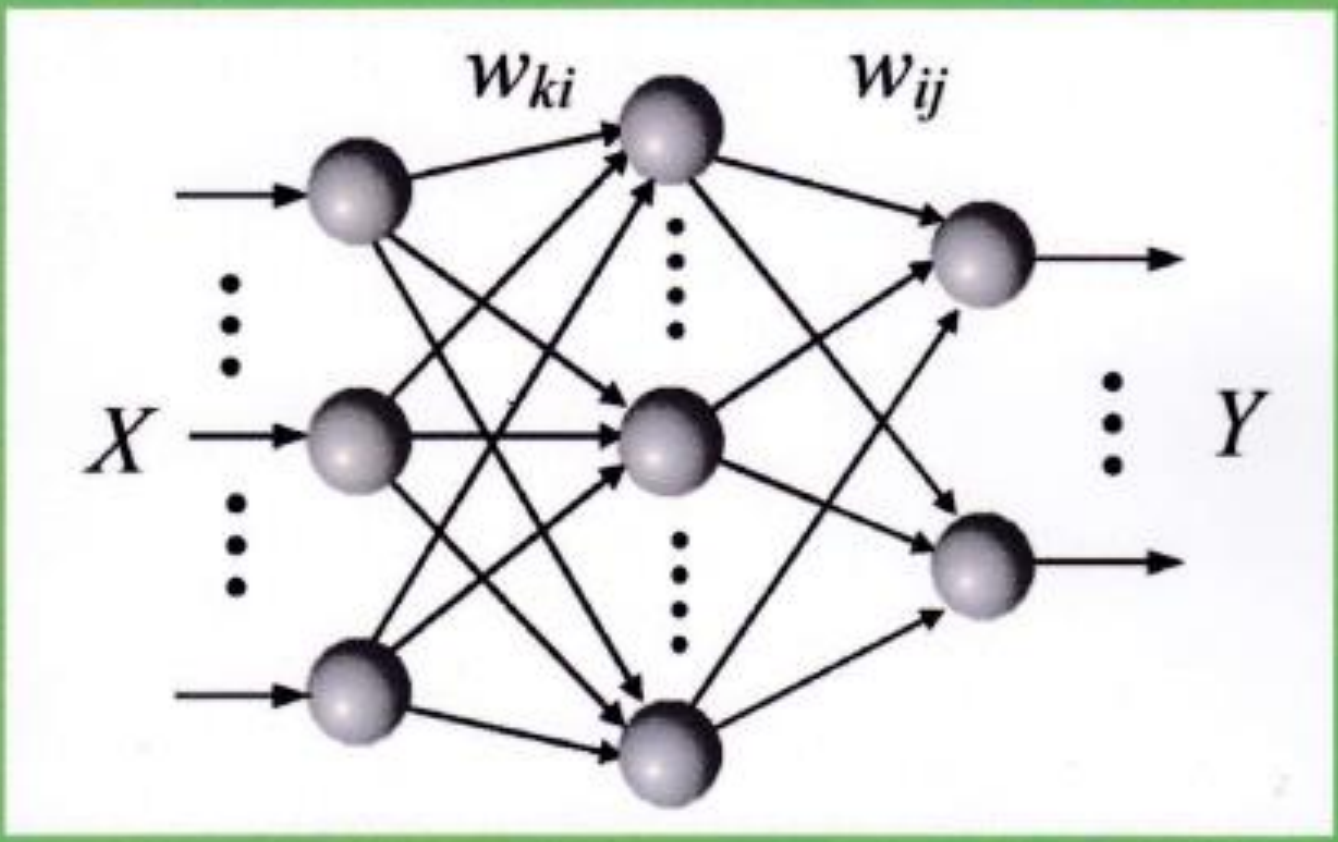
Міністерство освіти і науки України
Державний університет телекомунікацій

Звенігородський О.С., Катков Ю.І., Прокопов С.В.
Іщеряков, С.М., Рижаков М.М.

ШТУЧНИЙ ІНТЕЛЕКТ

Методичні вказівки до практичних занять з дисципліни «Штучний інтелект» для студентів спеціальності: 121 Інженерія програмного забезпечення, 123 Комп'ютерна інженерія, 124 Системний аналіз, 125 Кібербезпека, 126 Інформаційні системи та технології усіх форм навчання знань



Методичні вказівки до практичних занять з дисципліни "Штучний інтелект" для студентів спеціальності: 121 Інженерія програмного забезпечення, 123 Комп'ютерна інженерія, 124 Системний аналіз, 125 Кібербезпека, 126 Інформаційні системи та технології усіх форм навчання / Уклад.: А.С. Звенігородський, Ю.І. Катков – Київ: ДУТ, 2019. – 79 с.

Укладачі: Олександр Сергійович Звенігородський, к.т.н., доцент

Юрій Ігоревич Катков, к.т.н., доцент

Рецензент: Г.І. Гайдур, д.т.н., доцент

Затверджено на засіданні кафедри Протокол № 8 від 11 лютого 2019 р.

ЗМІСТ

Вступ

Завдання навчальної дисципліни „Штучний інтелект” – на основі отриманих теоретичних знань виробити у студентів уміння користуватися існуючими, а також створювати власні продукційні системи, системи нечіткої логіки, штучні нейронні мережі.

Метою проведення практичних занять є отримання навичок розробки та реалізації основних елементів систем штучного інтелекту. Перша частина методичних вказівок присвячена вивченню основних конструкцій та принципів програмування з використанням мови програмування Prolog. В другій частині розглядаються принципи створення нечітких моделей виведення і систем керування об’єктами засобами Fuzzy Logic Toolbox програмного пакету MATLAB. Третя частина присвячена моделювання штучних нейронних мереж різного типу в Neural Network Toolbox програмного пакету MATLAB.

У результаті вивчення дисципліни студенти повинні:

ЗНАТИ: основні методи подання знань, принципи нечіткого логічного виведення, структуру продукційних систем, структуру та принципи функціонування штучних нейронних мереж.

ВМІТИ: аналізувати і проектувати бази знань й продукційні системи, використовувати нечітке логічне виведення; створювати, навчати і використовувати штучні нейронні мережі.

Кожне практичне заняття містить: тему і мету заняття; опис обладнання та програмних засобів, потрібних для виконання даної лабораторної роботи; список літератури; короткі теоретичні відомості, необхідні для виконання завдання; порядок виконання завдання; контрольні запитання для самоперевірки.

Практичне заняття 1

Створення бази фактів і бази знань засобами SWI-Prolog.

Мета. Вивчити структуру продукційної системи, засвоїти основи мови програмування Prolog і порядок роботи в online-середовищі SWI-Prolog.

Завдання

1. Згідно з варіантом створити базу фактів і 1, 2 правила бази знань для предметної області (табл. 1).
2. Зв’язки предметної області подати у виді блок-схеми.
3. Створити SWI-Prolog програму.
4. Навести приклади п’яти типів запитань до створеної бази фактів і знань.
5. Проаналізувати отримані результати і зробити висновки.

Примітка. Номер варіанта співпадає з номером зі списку групи. У разі, коли номер зі списку перевищує кількість варіантів, варіант визначається як ццлочисельний залишок від ділення номера зі списку на кількість варіантів. Наприклад, якщо номер у списку – 31, то варіант – 9 (31/22 =1 залишок 9).

Таблиця 1 – Перелік варіантів предметної області

№ з/п	Предметна область
1.	Клуб за інтересами (прихильники письменників)
2.	Клуб за інтересами (любителі собак)
3.	Друзі і вороги
4.	Адміністративна структура компанії
5.	Генеалогічне дерево київських князів
6.	Структура спортивної команди
7.	Простір доменних імен DNS (Domain Name System)
8.	Адміністративна структура магазину.
9.	Класифікація музичних інструментів
10.	Класифікація спортивних ігор.
11.	Класифікація медичних препаратів
12.	Звання в армії.
13.	Тварини: хижаки і жертви
14.	Країни: столиця, населення
15.	Адміністративна структура ДУТ
16.	Навчальний процес: викладач, асистент, лекція, практика.
17.	Навчальний процес: дисципліна, викладач, група, студент
18.	Склад медичного персоналу лікарні
19.	Склад спортивної команди Динамо Київ.
20.	Компанії та продукти які вони випускають
21.	Адміністративна структура школи
22.	Адміністративна структура України

Зміст звіту

1. Титульна сторінка.
2. Тема практичного заняття.
3. Мета заняття.
4. Завдання.
5. Опис виконання завдань по пунктам з наданням блок-схем і скріншотів.
6. Текст SWI-Prolog програми
7. Висновки.

Контрольні питання.

1. Механізм логічного виводу в Prolog.
2. Типи предикатів в Prolog.
3. Типи запитань в Prolog.
4. Переваги і недоліки продукційних систем.

Теоретичні відомості

1. Основні поняття Prolog.

Програма на Prolog складається з речень (тверджень). Кожне речення закінчується крапкою. В загальному випадку речення складається із заголовка і тіла (предиката).

Речення має вигляд:

Речення бувають трьох видів: факти, правила, питання.

Факт – це речення, яке фіксує (визначає) певне відношення між об'єктами. Об'єкти називаються термами. Наприклад, факт, що Наталя є мамою Даші, може бути записаний таким чином:

mother ('Наталя', 'Софія').

Факт має тільки тіло, яке позначається ім'ям **mother** і два терми **'Наталя'** і **'Софія'**, зв'язані логічною операціє **I**, яка позначена символом **« , »**. Терми і зв'язок між ними подаються у круглих дужках. Факт може складатись і з одного терму. Факт є безумовно істинним твердженням. Термами факту можуть бути константа, змінна або складений об'єкт (список або функція). Як правило, об'єкт, що є причиною іншого об'єкта ставиться першим, але це не обов'язково. Важливо дотримуватись вибраної послідовності термів факту на протязі всієї програми. Факт є безумовно істинним твердженням.

Правило – речення, яке може бути істинним або не істинним, складається з заголовка і тіла. Істинність залежить від істинності однієї або декількох формул, зазначених в тілі. Зазвичай правило містить кілька цілей, які повинні бути істинними для того, щоб саме правило було істинним. Правило має ім'я, після якого йдуть символи **« :- »** – ознака правила.

B:- A1, ..., An.

B називається заголовком або головою речення, а **A1, ..., An** – тілом. У цьому реченні ключовою логічною операцією є перевернута імплікація (**B:- A** еквівалентно **B ← A**, «B випливає з A»). Символ "**:-**" означає "випливає з»,

символ «,» – кон'юнкція (логічне І, ∧).

При необхідності застосування диз'юнкції (логічне АБО, ∨) використовується символ «;», що діє до наступної диз'юнкції, закінчення правила або закриваючої круглої дужки, наприклад**, D:- A, (B; C)**. В логікі предикатів це виглядає так: **A ∧ (B ∨ C) → D.**

Наприклад, відомо, що бабуся людини – це мама його мами або мама його тата. Відповідні правила матимуть вигляд:

grandmother (X, Y):- mother (X, Z), mother (Z, Y).

grandmother (X, Y): - mother (X, Z), father (Z, Y).

або

grandmother (X, Y):- mother (X, Z), (mother (Z, Y); father (Z, Y)).

Питання (запит, мета) – речення, що складається тільки з тіла. Запити позначаються символами **«?-»**.

Запити використовують для з'ясування здійсненності деяких відносин між описаними в програмі об'єктами. Автоматична система логічного виведення Prolog розглядає питання як мету, яку треба довести. Відповідь на питання може виявитися позитивною (true) або негативною (false), в залежності від того, чи може бути досягнута мета.

Програма може містити питання в тілі (внутрішня мета). Якщо внутрішньої мети в програмі немає, то після запуску програми система видає запрошення **«?-»** вводити питання в діалоговому режимі (зовнішня мета). Якщо мета досягнута, система відповідає «yes» («true»), в іншому випадку «no» («false»). Слід зазначити, що відповідь «no» на питання не завжди означає, що вона негативна. Система може дати таку відповідь і в тому випадку, коли у неї просто недостатньо інформації, щоб позитивно відповісти на питання. Тобто Prolog заснований на так званій «Моделі закритого світу», в якій все, що можна отримати на основі опису моделі є істиною, а решта – хибність.

Заголовок в Prolog повинен бути складатись з літер латинського алфавіту і починатися з малої літери. Ім'я терму повинне починатись з малої літери латинського алфавіту, або поміщатись в одинарні лапки, наприклад, **'Наталя'**, тоді алфавіт може будь який.

Змінні. Ім'я змінної в Prolog може складатися з літер латинського алфавіту, цифр, знаків підкреслення і повинно починатися з великої літери або символу підкреслення. Наприклад, у реченні **grandmother (X, Y):- mother (X, Z), mother (Z, Y)** змінні мають імена з однієї літери – **X, Y. Z**. При цьому змінні в тілі правила еквівалентні об'єктам предметної області. Змінні можуть бути вільними або зв'язаними.

Вільна змінна – змінна, яка ще не отримала значення. Вона не дорівнює ні нулю, ні пробілу; у неї взагалі немає ніякого значення. Такі змінні ще називають неконкретизованими.

Змінна, яка отримала якесь значення, називається зв'язаною. Такій змінній не може бути присвоєно нове значення.

Областю дії змінної в Prolog є одне речення. У різних реченнях може використовуватися одне й теж ім'я змінної для позначення різних об'єктів. Винятком з правила визначення області дії є анонімна змінна, яка

позначається символом підкреслення «_». Анонімна змінна наказує інтерпретатору (компілятору) проігнорувати значення терму. Анонімні змінні можуть записуватися тільки в якості терму предиката. Використовувати їх у виразах (наприклад, арифметичних) не можна.

Таким чином, програма на Prolog складається з фактів і правил, що виражають деякі знання про предметну область. Питання – це предикат, істинність якого нас цікавить. Якщо питання не містить змінних, то обчислення його значення дає відповідь «true» при його істинності, або відповідь «false» при його хибності. Якщо в питанні є змінні, то відшукуються їхні значення, при яких цей предикат і всі предикати програми стають істинними. В цьому і полягає суть логічного виведення програми на Prolog.

2. Приклад бази фактів і бази знань в Prolog

Розглянемо приклад бази фактів і бази знань сімейних відносин. Зауважимо, що символ % означає, що після нього йде коментар, який не впливає на програму.

```
% База фактів сімейних відносин
mother ('Наталя', 'Софія').
mother ('Софія', 'Марія').
mother ('Наталя', 'Василь').
father ('Василь', 'Марія').
% База правил (знань) експертної системи сімейних відносин
grandmother (X, Y):-
mother (X, Z), (mother (Z, Y); father (Z, Y)).
grandfather (X, Y):-
father (X, Z), (mother (Z, Y); father (Z, Y)).
```

Розглянемо, як питання природною мовою записуються в Prolog.

Питання 1. Чи є Наталя матір'ю Софії?

```
?-mother ('Наталя', 'Софія').
true.        % Відповідь системи.
```

Відповідь означає, що такий факт в базі фактів існує (є істинним), тобто Наталя є матір'ю Софії.

Питання 2. Хто є матір'ю Софії?

Для таких питань необхідно використовувати змінні, наприклад, змінну з ім'ям X. Змінна X отримує значення першого терму предикатів-фактів **mother**, в яких другим термом є терм **'Софія'**

```
?-mother(X, 'Софія').
X = 'Наталя';
false.
```

Перший рядок повідомлення означає, що відповідь знайдена і матір'ю Софії є Наталя. Другий – що в базі знань для решти речень не виявлені інші матері Софії.

Питання 3 – Чи є у Софії матір? Використовується анонімна змінна.

```
?-mother(_, 'Софія').
```

true.

Питання 4 – Знайти всіх матерів і дітей.

?-mother(X, Y).
X = 'Наталя',
Y = 'Софія';
X = 'Софія',
Y = 'Марія';
X = 'Наталя',
Y = 'Василь'.

В даному випадку після третьої відповіді не видається **false**, тому що в базі знань були перебрані всі речення і вони всі істинні. Як виглядає відповідь на це питання в online **SWI-Prolog** див. рис.4.

Питання 5 – Для кого Наталя є бабусею?

?-grandmother('Наталя', X).
X = 'Марія';
X = 'Марія'.

В даному випадку видається дві однакові відповіді «Марія», тому що правило grandmother в одному випадку спрацювало по ланцюжку «Наталя – Софія – Марія», а в іншому – «Наталя – Василь – Марія». Очевидно, що в наведеному прикладі бази знань бабусі **'Наталі'** – це дві різні жінки з однаковими іменами і онучки теж мають однакові імена **'Марія'**.

3. Процедура виведення в Prolog

При пошуку розв'язку (доказу мети) в Prolog використовується метод перебору з поверненнями (з пошуком в глибину). Prolog при доказі твердження по черзі намагається встановити істинність предикатів (тверджень), що входять в нього. Якщо перший предикат істинний, то Prolog переходить до другого. Якщо і він істинний, то переходить до третього. Якщо другий предикат хибний, то Prolog намагається встановити його істинність при інших значеннях змінних, що входять в нього. Якщо цього не вдається зробити, то він повертається до першого предикату і намагається встановити його істинність для нових значень змінних, а потім знову повертається до доказу другого предиката. Така процедура повторюється до тих пір, поки не буде досягнута істинність останнього предиката. Після доведення істинності останнього предиката мети Prolog завершує роботу. Процес повернення в Prolog називається **backtracking**.

Приклад виведення.

Визначимо питання 5 про онучок і онуків «Наталі» більш детально:

?- **mother('Наталя', Y),**
(mother (Y, Z); father (Y, Z)),
write (Z);
Марія ; % відповідь системи
Марія ;

Примітка. Частина правила з підкресленням відповідає правилу визначення бабусі (grandmother).

Процедура виведення (перебору з поверненнями) для цього прикладу наведена в табл. 2.

Таблиця 2 – Приклад виведення в SWI-Prolog

№ з/п	Предикат запиту	Предикат бази знань, що перевіряється	Результат
1	**mother('Наталя', Y)**	**mother('Наталя', 'Софія')**	**Y = 'Софія'**
2	mother(Y, Z) ≡ mother ('Софія', Z)	mother('Наталя', 'Софія')	backtracking
3	**mother(Y, Z) ≡ mother ('Софія', Z)**	**mother('Софія', 'Марія')**	**Z = 'Марія'**
4	**write(Z) ≡ write('Марія')**		**'Марія'**
5	mother(Y, Z) ≡ mother('Софія', Z)	mother('Наталя', 'Василь')	backtracking
6	father(Y, Z) ≡ father ('Софія', Z)	father('Василь, 'Марія')	backtracking
7	mother('Наталя', Y)	mother('Софія', 'Марія')	backtracking
8	**mother('Наталя', Y)**	**mother('Наталя', 'Василь')**	**Y ='Василь'**
9	mother(Y, Z) ≡ mother('Василь', Z)	mother('Наталя', 'Софія')	backtracking
10	mother(Y, Z) ≡ mother('Василь', Z)	mother('Софія', 'Марія')	backtracking
11	mother(Y, Z) ≡ mother('Василь', Z)	mother('Наталя', 'Василь')	backtracking
12	**father(Y, Z) ≡ father('Вася', Z)**	**father('Василь', 'Марія')**	**Z = 'Марія'**
13	**write(Z) ≡ write('Марія')**		**'Марія'**

Примітка. Жирним виділені рядки, для яких виконання предиката істинно.

4. Програмний засіб SWI-Prolog

SWI-Prolog є вільно розповсюджуваною (Free Software) реалізацією (діалектом) мови програмування Prolog, яка практично повністю відповідає стандарту ISO/ EC13211.

SWI-Prolog дозволяє розробляти програми будь-якої спрямованості, включаючи Web-додатки і паралельні обчислення, але основним напрямом використання є розробка експертних систем, програм обробки природної мови, навчальних програм, інтелектуальних ігор тощо.

Програмування в SWI- Prolog можливо в різних варіантах:

- за допомогою стандартного offline-середовища у вільному доступі, можна завантажити з (http://www.swi-prolog.org кнопка **Download SWI-Prolog**

- за допомогою online-середовища на сайті http://www.swi-prolog.org кнопка **Try SWI-Prolog online**

Для переходу в режим редагування і виконання програм необхідно натиснути на кнопку «Program» (рис.2). Після цього можна створювати і редагувати Prolog-програму, створювати запити і отримувати відповіді (рис.3).

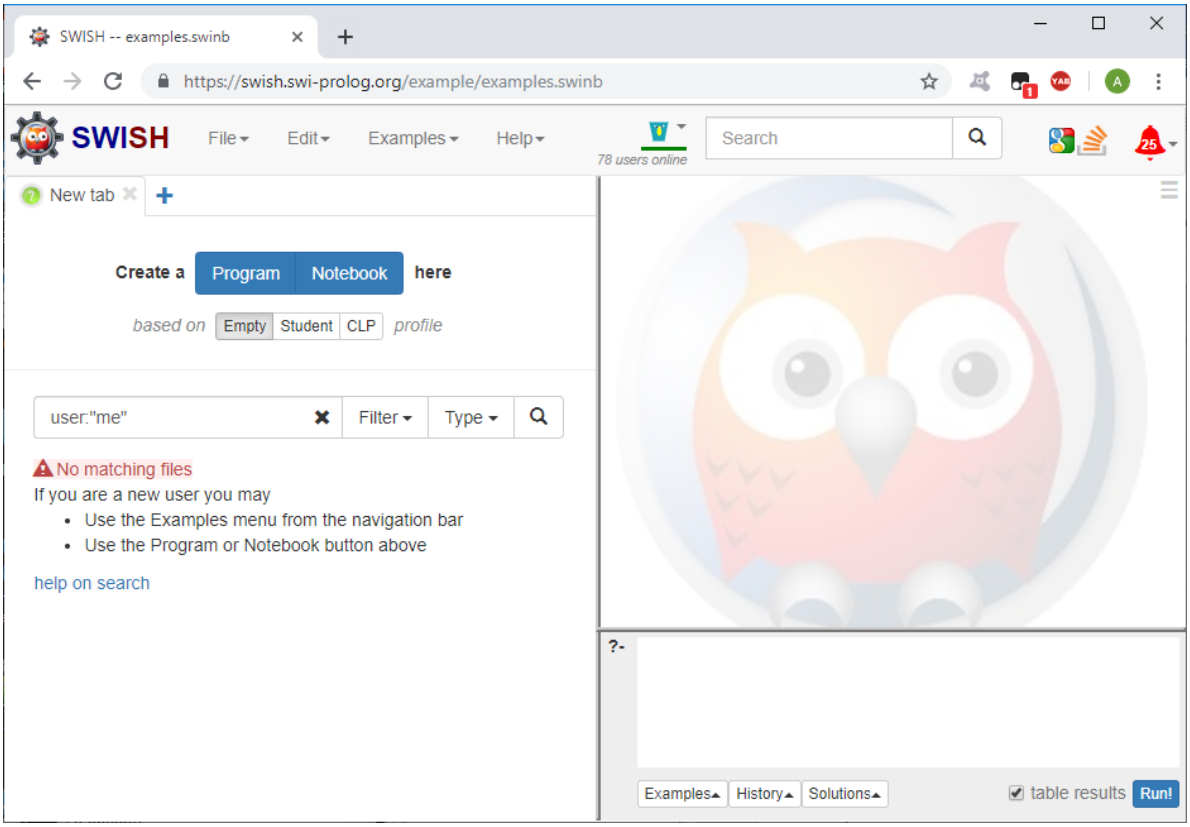


Рисунок 2 – Стандартне online-середовище програмування SWI-Prolog

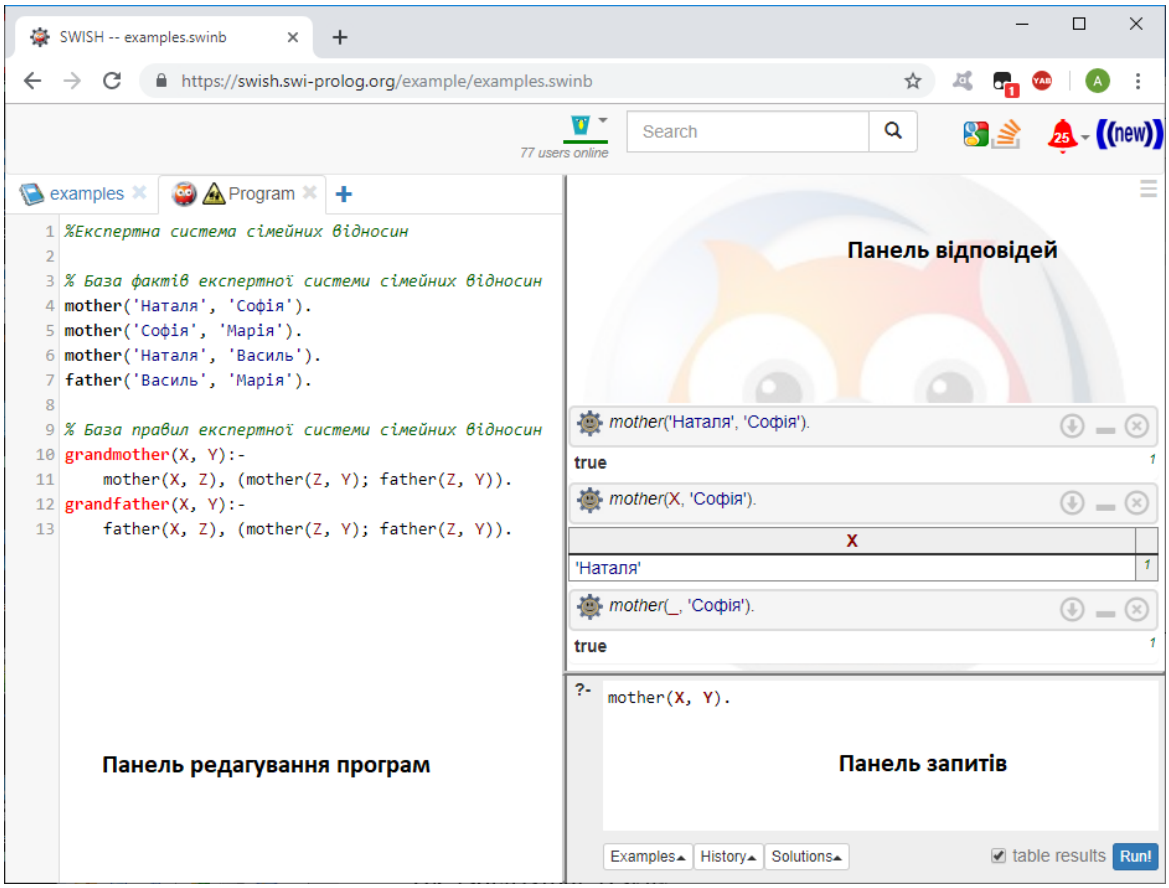


Рисунок 3 – Режим редагування і виконання програм

У лівій панелі здійснюється редагування програми, яка містить факти і правила.

У правій нижній панелі виконується набір питань і запуск їх на виконання за допомогою кнопки «Run!».

У правій верхній панелі інтерпретатор SWI-Prolog видає відповіді на запити. У разі якщо на питання може бути отримано більше однієї відповіді, за допомогою кнопок «Next», «10», «100» і «1,000» можна вивести на панель інші відповіді.

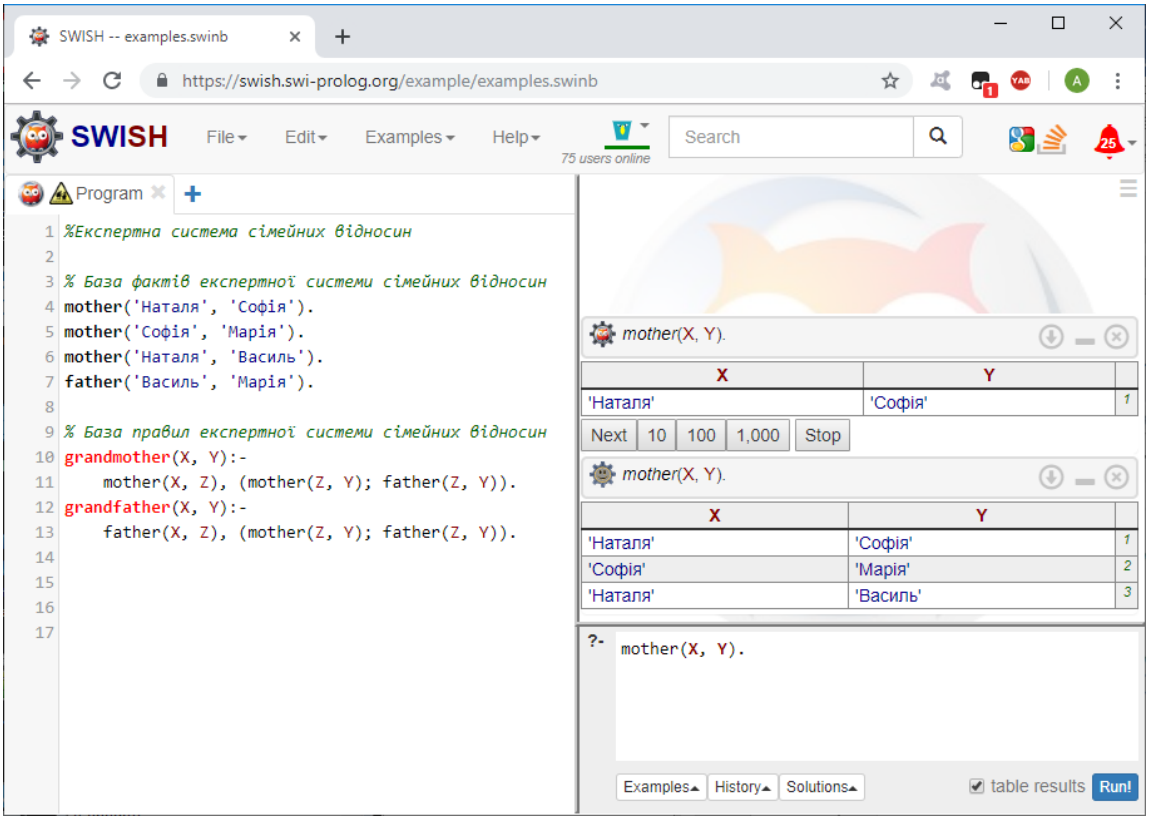


Рисунок 4 – Застосування кнопок «Next», «10», «100» і «1,000»

Перелік рекомендованої літератури

1. Субботін С.О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень : навчальний посібник / С.О. Субботін. – Запоріжжя: ЗНТУ, 2008. – 341 с.

2. Братко И. Программирование на языке Пролог для искусственного интеллекта Пер. с англ. – М.: Мир, 1990. - 560 с., ил.

3. Джарратано, Д. Экспертные системы: принципы разработки и программирование / Джарратано, Джозеф, Райли, Гари., 4-е издание.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2007. – 1152 с.: ил.

4. Логика предикатов первого порядка. – [Електронний ресурс]: [Веб-сайт] - Електронні дані. Режим доступу: https://sites.google.com/site/anisimovkhv/learning/iis/lecture/tema9 – Назва з екрана

5. Язык логического программирования Пролог. – [Електронний ресурс]: [Веб-сайт] – Електронні дані. Режим доступу: https://sites.google.com/site/anisimovkhv/learning/iis/lecture/tema10 – Назва з екрана

Практичне заняття 2

Тема: Створення експертної системи засобами SWI-Prolog.

Мета. Вивчити склад і основні принципи побудови експертних систем на засадах логічних моделей подання знань та навчитися будувати інтерфейс користувача експертної системи.

Завдання

1. В якості предметної області обрати предметну область згідно з варіантом (табл. 1), або з варіантом практичного заняття 1.
2. Визначити мету, задачі експертної системи та питання користувача до експертної системи.
3. Визначити правила бази знань.
4. Побудувати блок-схему (дерево) логічних можливостей для вибору.
5. Побудувати схему структуру діалогу з користувачем.
6. Створити базу знань за допомогою засобів SWI-Prolog.
7. Проаналізувати отримані результати і зробити висновки.

Примітка. Номер варіанта співпадає з номером зі списку групи. У разі, коли номер зі списку перевищує кількість варіантів, варіант визначається як цілочисельний залишок від ділення номера зі списку на кількість варіантів. Наприклад, якщо номер у списку – 31, то варіант – 9 (31/22 =1 залишок 9).

Таблиця 1 – Перелік рекомендованих предметних областей для створення експертної системи.

№ з/п	Предметна область
1.	Індивідуальний підбір косметики
2.	Індивідуальний підбір одягу
3.	Індивідуальний підбір взуття
4.	Індивідуальний підбір мобільного телефону
5.	Індивідуальний підбір музичного центру
6.	Індивідуальний підбір відеокамери
7.	Індивідуальний підбір телевізора
8.	Індивідуальний підбір комп’ютера
9.	Індивідуальний підбір автомобіля
10.	Індивідуальний підбір житла
11.	Індивідуальний підбір спеціальності для абітурієнтів
12.	Класифікація птахів
13.	Класифікація квітів
14.	Класифікація риб
15.	Класифікація дерев
16.	Класифікація овочів
17.	Класифікація собак
18.	Класифікація мавп
19.	Класифікація фруктів
20.	Класифікація комах
21.	Діагностика несправності автомобіля

Зміст звіту

1. Титульна сторінка.
2. Тема практичного заняття.
3. Мета заняття.
4. Завдання.
5. Опис виконання завдань по пунктам з наданням блок-схем, рисунків і скріншотів.
6. Текст SWI-Prolog програми
7. Висновки.

Контрольні питання

1. Формальна модель продукційної системи подання знань.
2. Структура статичної експертної системи.
3. Етапи створення експертної системи.
4. Режими роботи експертних систем.
5. Механізм логічного виводу в Prolog.
6. Учасники розробки експертної системи.

Теоретичні відомості.

1. Подання знань в продукційних системах

Продукція визначається як четвірка:

$$< (i)\ Q;\ P;\ A \Rightarrow B;\ N >,$$

де:

(i) – ім’я продукції;

Q – предметна область (сфера застосування);

P – умова застосування продукції;

$A \Rightarrow B$ – ядро продукції (правило) інтерпретується як «**Якщо маєш А, зроби В**» або «**Якщо А то В**»;

N – післяумова продукції (процедури, які необхідно виконати після реалізації ядра.

Найбільше застосування ця модель знайшла в продукційних (логічних) експертних системах (рис. 1). Для спрощення реалізації таких систем створена мова програмування Prolog. Її особливістю є те, що алгоритм логічного виведення вбудований в неї, що не вимагає писати програму логічного виведення, чим прискорює розробку і тестування систем.

У математичних термінах Prolog-програма інтерпретується наступним чином:

- факти і правила – множина аксіом;
- питання користувача – теорема;
- механізм виведення намагається з аксіом логічно довести цю теорему (в Prolog ця теорема називається метою).

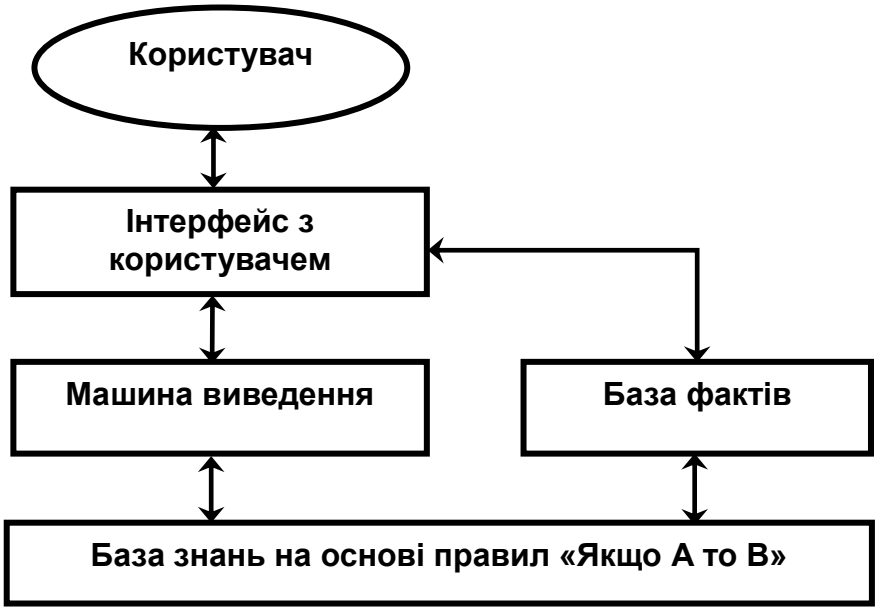


Рисунок 1 – Структура продукційної експертної системи

Факти і правила використовуються для доведення мети. Prolog шукає факти і заголовки правил, зіставні з метою. Факт із бази фактів і факт з умови правила відповідають один одному, якщо виконуються наступні три умови: імена фактів однакові; факти мають рівну кількість термів; терми розташовані в однакових позиціях.

2. Етапи створення експертної системи.

Ідентифікація. Необхідно визначити мету і задачі експертної системи; визначити експертів і тип користувачів.

Концептуалізація. Проводиться змістовний аналіз предметної області; виділяються основні поняття і їх взаємозв'язки; визначаються методи рішення задач.

Формалізація. Вибираються програмні засоби розробки ЕС, визначаються способи представлення усіх видів знань, формалізуються основні поняття.

Виконання. Здійснюється наповнення бази знань. Знання "витягаються" з експерта, представляються у логічному або інших видах для реалізації програмно.

Тестування. Експерт і інженер по знанням з використанням діалогових і пояснювальних засобів перевіряють компетентність ЕС. Процес тестування продовжується до визнання експертом необхідного рівня компетентності системи.

Дослідна експлуатація. Перевіряється придатність ЕС для кінцевих користувачів**.**

3. Режими роботи експертних систем

Експертні системи працюють у двох режимах:

- придбання знань;
- режим рішення задачі (режим консультації).

У режимі придбання знань інженер по знанням по результатам спілкування з експертом наповнює ЕС новими знаннями і фактами.

У режимі консультації користувач взаємодіє з ЕС для рішення своєї задачі шляхом діалогу. Якщо в системі не вистачає фактів для доведення мети, наприклад встановлення діагнозу, то вона задає питання користувачеві і той ці факти надає. Кількість запитань залежить від внутрішньої структури діалогу конкретної ЕС. Після закінчення діалогу машина виведення доводить або спростовує мету.

4. Модель експертної системи для вибору туристичної поїздки

(Приклад розробки)

Етап ідентифікації експертної системи.

Постановка завдання: у процесі вибору туру важко зупинитися на якійсь моделі, тому що велика кількість фірм надає подібні послуги з різними характеристиками, і в цьому достатньо важко орієнтуватися. Пропонується розробити прототип експертної системи (ЕС) прийняття рішень по вибору туру за перевагами, які є важливими для клієнта (користувача).

Призначення ЕС: консультування клієнта під час визначення оптимального туру за властивостями, яким він надає перевагу.

Сфера застосування прототипу ЕС: туристичні фірми, що займаються продажом турів.

Ціль: вибір оптимального варіанта туру для клієнта згідно з його потреб і вимог до послуги.

Вхідні дані: країна відвідування, вид відпочинку, тривалість відпочинку.

Очікувані результати: конкретний тур.

Об’єкти (фактори) Предметної Області (ПрО). Тур – це послуга, на вибір якої впливають різні фактори. Виходячи з соціологічного опитування, найбільш важливими факторами вибору є наступні:

- країна відвідування;
- вид відпочинку;
- тривалість відпочинку.

Етап концептуалізації експертної системи.

Кількість об’єктів ПрО – 6. Дерево логічних можливостей вибору туру подане на рис. 1.

Загальна кількість можливих варіантів турів дорівнює добутку всіх кількостей значень атрибутів (3 х 3 х 2 = 18). Але серед цих варіантів експерт відібрав тільки 6, які відповідають існуючим на даний час пропозиціям.

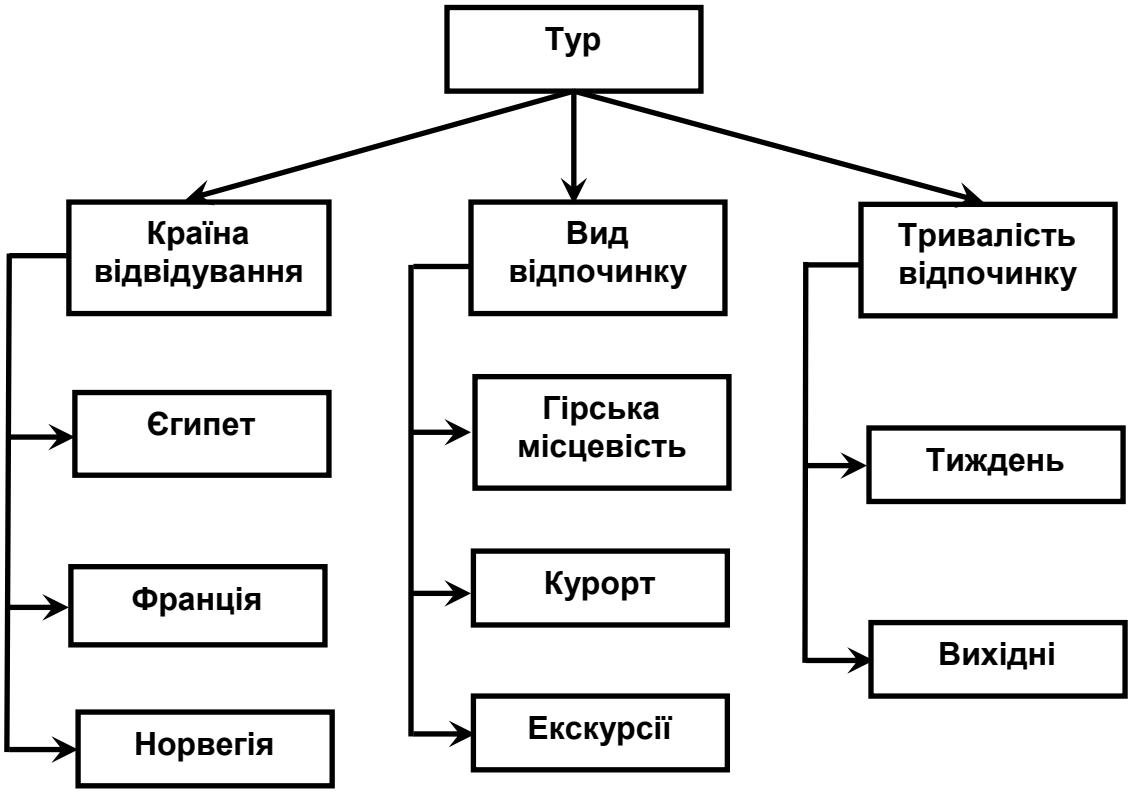


Рисунок 1 – Дерево логічних можливостей для вибору туру

Таблиця 2 – Атрибути Бази знань

Атрибут (термін)	Питання (опис)	Відповідь
Країна	Яку країну Ви бажаєте відвідати?	egypt
		france
		norway
Тип відпочинку	Якому виду відпочинку Ви надаєте перевагу?	mountain
		resort
		sightseeing
Тривалість відпочинку	Як довго Ви хочете відпочивати?	weekend
		week

Наведемо приклад правила:

ЯКЩО

Країна = egypt **I**
Тип Відпочинку = mountain **I**
Тривалість відпочинку = weekend

ТО

Тур = 'Квадро-тур на Синай'.

5. Операції та вбудовані предикати SWI-Prolog

У табл. 3 наведені деякі операції і предикати SWI-Prolog, які в подальшому будуть використовуватись для ілюстрації прикладів.

Таблиця 3 – Деякі операції та предикати SWI-Prolog

Операція / Предикат	Призначення
true	істина
fail, false	Неправда, хибність
=	• вільна змінна – присвоювання без обчислення виразу праворуч від операції; • зв'язана змінна – порівняння без обчислення виразу праворуч від операції.
<, =<, >=, >	Арифметичні (тільки для чисел) операції порівняння
=:=	арифметична рівність
=\=	арифметична нерівність
is	• вільна змінна – присвоювання з обчисленням виразу; • зв'язана змінна – порівняння з обчисленням виразу.
@<, @=<, @>=, @>	Операції порівняння для констант і змінних будь-якого типу (чисел, рядків, списків і т.д.)
==	Рівність констант і змінних будь-якого типу
\==	Нерівність констант і змінних будь-якого типу
not(A)	Заперечення логічного виразу A
read(A)	Читання значення з клавіатури і присвоювання його змінній A
write(A)	Друк A на екрані з установкою курсору після останнього надрукованого символу
writeln(A)	Друк A на екрані з установкою курсору в початок наступного рядка
nl	Установка курсора в початок наступного рядка
!	Предикат (cut, скоротити), що забороняє повернення далі тієї точки, де він стоїть
assert(A), assertz(A)	Динамічне додавання факту (правила) в початок списку подібних фактів (правил) бази фактів і знань
asserta(A)	Динамічне додавання факту (правила) в кінець списку подібних фактів (правил) бази фактів і знань
retract(A)	Видалення першого факту (правила) бази фактів і знань
retractall(A)	Видалення всіх фактів (правил) бази фактів і знань з ім'ям A
dynamic(A/n)	Вказівка, що предикат A може змінюватись в процесі виконання програми за допомогою предикатів **assert і retract**, n – кількість термів предикату A

6. Програмна реалізація експертної системи з вибору туру

На початку діалогу в базі фактів факти відсутні. Під час діалогу користувач вводить факти і після цього робиться логічне виведення. Для організації діалогу застосовуються вбудовані предикати **write(A)** – вивід на екран, **nl –** перехід на новий рядок на екрані, **read(A)** – читання з клавіатури, **assert(A)** – динамічне додавання факту в базу фактів (дозвіл на зміну бази фактів дає предикат **:- dynamic**. Система задає три питання: **ask1, ask2, ask3**.

```
main:-
        greeting,
        ask1, ask2, ask3,
        find_tour(Product),
        describe(Product),
   nl.

greeting :-
        write('Експертна система з підбору туристичної путівки'), nl, nl.

:- dynamic(country/1).
:- dynamic(type/1).
:- dynamic(duration/1).

ask1:- write('Яку країну Ви бажаєте відвідати?)', nl,
write (egypt, france, norway), nl,
  read(N), assert(country(N)).

ask2:- write('Якому виду відпочинку Ви надаєте перевагу?'), nl,
write(mountain, resort, sightseeing), nl,
  read(N), assert(type(N)).

ask3:-write('Як довго Ви хочете відпочивати?'), nl,
write(weekend, week), nl,
   read(N), assert(duration(N)).

find_tour(Tour) :-
        tour(Tour), !.
                                    %База знань (правил)
tour('Квадро-тур на Синай') :-
  country(egypt),
  type(mountain),
  duration(weekend).

tour('Піший похід на гору Мойсея') :-
   country(egypt),
   type(mountain),
   duration(week).

tour('Вікенд в Хургаді') :-
   country(egypt),
   type(resort),
   duration(weekend).
```

```
tour('Тиждень в Шарль-Ем-Шейху') :-
    country(egypt),
    type(resort),
    duration(week).

tour('Похід на Монт Пелат') :-
    country(france),
    type(mountain),
    duration(weekend).

    tour('Екскурсійний тур по Осло') :-
    country(norway),
    type(sightseeing),
    duration(weekend).
      % Вивід результату коли яке-небудь правило стане істинним (спрацює)
describe(Tour):-
write('Тур, що Вам підходить '), nl,
    write(Tour), nl, nl.
```

Для запуску програми необхідно на панелі запитів SWI-Prolog набрати **main** і натиснути **Run!**

Перелік рекомендованої літератури

1. Хабаров С.П. Интеллектуальные информационные системы. PROLOG- язык разработки интеллектуальных и экспертных систем: учебное пособие / С.П. Хабаров. – СПб. СПбГЛТУ, 2013. – 138 с.
2. Разработка экспертных систем в Prolog – [Електронний ресурс]: [Веб-сайт] - Електронні дані. Режим доступу: http://itmu.vsuet.ru/Posobija/Predstavlenie_znan/htm/5_pr.htm – Назва з екрана
3. Логика предикатов первого порядка. – [Електронний ресурс]: [Веб-сайт] - Електронні дані. Режим доступу: https://sites.google.com/site/anisimovkhv/learning/iis/lecture/tema9 – Назва з екрана
4. Язык логического программирования Пролог. – [Електронний ресурс]: [Веб-сайт] – Електронні дані. Режим доступу: https://sites.google.com/site/anisimovkhv/learning/iis/lecture/tema10 – Назва з екрана

Практичне заняття 3

Тема: Основи програмування в системі MATLAB

Мета. Вивчити основи проблемно-орієнтованої системи програмування MATLAB, типи даних, матричні операції, способи створення функцій, придбати навички роботи з командно-графічним інтерфейсом системи.

Завдання

1. Згідно з варіантом (табл. 1) розробити файл сценарію обчислення функції.
2. Згідно з варіантом (табл. 1) розробити файл функцію.
3. Згідно з варіантом (табл. 1) розробити файл функцію з введенням даних користувачем.

Примітка. Номер варіанта співпадає з номером зі списку групи. У разі, коли номер зі списку перевищує кількість варіантів, варіант визначається як цілочисельний залишок від ділення номера зі списку на кількість варіантів. Наприклад, якщо номер у списку – 31, то варіант – 9 (31/22 =1 залишок 9).

Таблиця 1 – Функції для реалізації

№ з/п	Функція файлу-функції	Функція файлу-сценарію
1.	$\cos^2(t)\sin(2t)$	Обчислення площі кола
2.	$\sin(0.5t^2)$	Обчислення площі прямокутника
3.	$t\sin(t)$	Обчислення периметру прямокутника
4.	$\cos(1.4^t)$	Обчислення довжини кола
5.	$\ln(t)\sin(t)$	Обчислення об'єму циліндра
6.	$\ln(2t)\sin(t^{1.5})$	Обчислення об'єму конуса
7.	$0.01t^2\sin(t)$	Обчислення площі сектора кола
8.	$t\sin(2t)$	Обчислення об'єму шару
9.	$t\sin(3t)$	Обчислення площі трапеції
10.	$t\sin(4t)$	Обчислення об'єму паралелепіпеду
11.	$t^2\sin(t)$	Обчислити факторіал
12.	$\sin(t^2-10t)$	Пошук найбільшого елементу вектору
13.	$\sin(t^2-5t)$	Пошук найменшого елементу вектору
14.	$\sin(t^2-8t)$	Обчислення суми елементів
15.	$\sin(t^2-12t)$	Обчислення суми квадратів елементів
16.	$\sin(t^2-8t)$	Обчислення sin через функцію e
17.	$\sin(t^2-12t)$	Обчислення cos через функцію e
18.	$\sin(t^2-6t)$	Обчислення tg через функцію e
19.	$\sin(t^2-4t)$	Обчислення ctg через функцію e
20.	$(t^2-10t)\sin(t^2-10t)$	Обчислення гіпотенузи прямокутного

Зміст звіту

1. Титульна сторінка.
2. Тема практичного заняття.
3. Мета заняття.
4. Завдання.
5. Опис виконання завдань по пунктам з наданням блок-схем, рисунків і скріншотів.
6. Тексти m-файлів
7. Висновки.

Контрольні питання

1. Для чого в MATLAB в кінці рядка використовується символ (;)?
2. Чим в MATLAB відрізняються команди (*) і (.*)?
3. У чому різниця між скалярним значенням, матрицею і вектором в MATLAB
4. Чим відрізняються файл-функція і файл-сценарій

Теоретичні відомості.

Потужна інструментальна система MATLAB забезпечує процедурне, функціональне, логічне, структурне, об'єктно-орієнтоване і візуальне програмування. Вона базується на математико-орієнтованій мові надвисокого рівня, яка спрощує запис алгоритмів і відкриває нові методи їх створення. Мова системи MATLAB за своєю структурою є командною. Команди мови виконуються в режимі інтерпретації. З її допомогою можна створювати текстові модулі-функції та модулі-сценарії. Файли, де зберігаються такі модулі, мають розширення ***.m** і називаються М-файлами, а функції, що в них знаходяться – М-функціями. У системі є величезна бібліотека М-функцій в текстовому форматі, які можна модифікувати для досягнення бажаних цілей. Користувач може створювати власні М-функції і включати їх в систему.

Вікно команд

MATLAB створений таким чином, що будь-які (іноді дуже складні) обчислення можна виконувати в режимі прямих обчислень, тобто без підготовки програми користувачем. При цьому MATLAB виконує функції Суперкалькулятора і працює в режимі командного рядка.

Робота з системою носить діалоговий характер і відбувається за правилом «задав питання – отримав відповідь ». Користувач набирає на клавіатурі враз, що обчислюється, редагує його (якщо потрібно) в командному рядку і на введення натисканням клавіші ENTER. Вікно команд і інші вікна MATLAB пре дставлені рис.1.

Браузер файлової структури

Для перегляду файлової структури MATLAB служить спеціальний браузер файлової системи (Path Browser), який запускається при звичайному завантаженні системи. Якщо був встановлений спрощений інтерфейс, то для запуску браузера файлової системи використовується вікно Current Folder. На

рис. 1 в лівій частині показано вікно цього браузера.

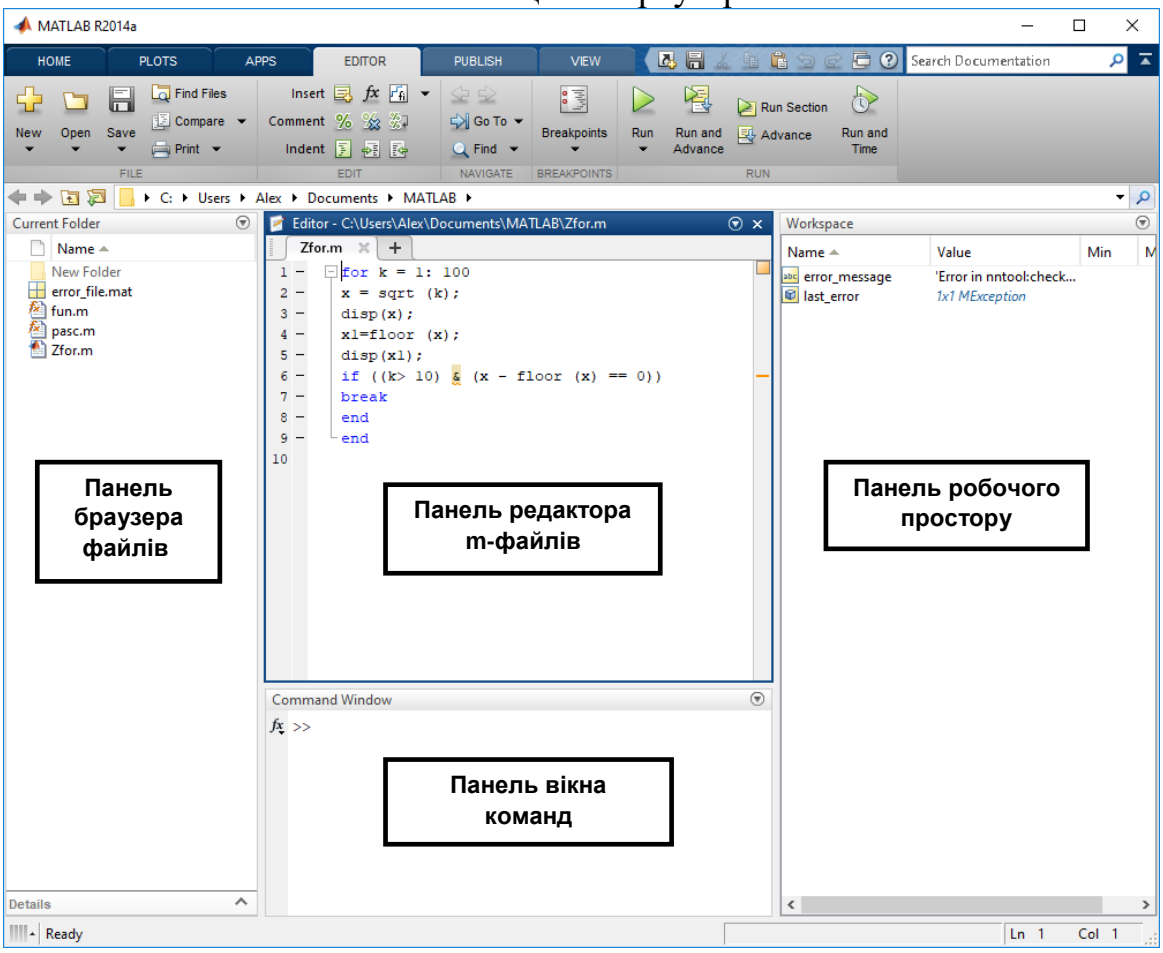


Рисунок 1 – Вікна MATLAB

Базові засоби мови програмування

Система MATLAB може використовуватися в двох режимах: в режимі безпосереднього обчислення (командний режим) і в режимі програмування.

У командному режимі користувач послідовно вводить команди вхідної мови і отримує відповідь. За допомогою цього режиму можна розв’язувати математичні задачі різної складності.

Однак при вирішенні серйозних завдань виникає необхідність збереження використовуваних послідовностей обчислень, а також їх подальших модифікацій. Для вирішення цього завдання система MATLAB має у своєму складі потужну мову програмування високого рівня.

Програмами в системі MATLAB є файли текстового формату з розширенням **.m**, що містять запис програм у вигляді програмних кодів.

Для редагування файлів програм може використовуватися будь-який текстовий редактор, а також спеціальний багатовіконний редактор. Редактор програм системи MATLAB має наступні можливості:

- колірне підсвічування синтаксису, що дозволяє виявити помилки;
- синтаксичний контроль на стадії підготовки М-файлу;
- установка точок переривання при інтерпретації команд;
- автоматична нумерація рядків програми для видачі повідомлень.

Мова програмування системи MATLAB має наступні засоби для побудови команд і написання М-файлів:

1) дані різного типу: double, numeric, char, cell, array;
2) константи і змінні: 25, pi, eps, ‘Hello’, ans, m, n;
3) оператори, включаючи оператори математичних виразів: +, -, *;
4) вбудовані команди і функції: help, clear, plot, sin, cos;
5) функції користувача: func, map, draw, paint, neuron;
6) оператори циклів: if, for, while, switch, try, catch, end;

Основи редагування та налагодження m-файлів

Інтерфейс редактора m-файлів

Для підготовки, редагування та налагодження m-файлів служить спеціальний багатовіконний редактор. Він виконаний як типовий додаток Windows. Редактор можна викликати командою edit з командного рядка або з меню New -> Script. Після цього у вікні редактора можна створювати свій файл, користуватися засобами його налагодження і запуску.

На рис. 1 показано вікно редактора з текстом простого файлу у вікні редагування та налагодження програмного коду.

Призначення кнопок панелі інструментів редактора:

- New – створення нового m-файлу;
- Open – завантаження нового файлу;
- Save – запис файлу на диск;
- Print – друк вмісту поточного вікна редактора;
- Breakpoints – установка / скидання точок переривання;
- **Run** – запуск програми на виконання і збереження;

Підготовлений текст файлу треба записати на диск. Для цього використовується меню **Save**. Після запису файлу на диск можна помітити, що команда **Run** в меню редактора стає активною (до запису файлу на диск вона пасивна) і дозволяє провести запуск файлу. Запустивши команду **Run**, можна спостерігати виконання m-файлу.

Для зручності роботи з редактором рядки програми в ньому нумеруються в послідовному порядку. Редактор є багато віконним. Вікно кожної програми оформляється як вкладка.

Кольорові виділення і синтаксичний контроль.

Редактор m-файлів виконує синтаксичний контроль програмного коду під час введення тексту. При цьому використовуються наступні колірні виділення:

- ключові слова мови програмування – синій колір;
- оператори, константи і змінні – чорний колір;
- коментарі після символу % – зелений колір;
- символьні змінні (в апострофа) – зелений колір;
- синтаксичні помилки – червоний колір.

Завдяки кольоровим виділенням ймовірність синтаксичних помилок знижується.

Робота з точками переривання

Основним прийомом налагодження m-файлів є установка в їх тексті точок переривання (Breakpoints), коли програма виконується до цієї точки і

зупиняється. В такому режимі ми можемо подивитись на результати, що досягнуті на цей момент, наприклад, поточні значення змінних і зробити висновок про правильність чи не правильність роботи програми, поті запустити далі до нової точки переривання і так до завершення програми. Точки переривання встановлюються (і скидаються) в меню **Breakpoints** за допомогою кнопок **Set/Clear**. Скидання всіх точок переривання забезпечується кнопкою **Clear All**.

Математичні обчислення

MATLAB – це математичний пакет прикладних програм, заснований на використанні матриць. Пакет містить велику бібліотеку програм з чисельних методів, використовує дво- і тривимірну графіку, а також формати мов високого рівня.

Арифметичні операції

\+ Додавання
\- Віднімання
* Добуток
/ Ділення
^ Піднесення до степеня
pi, e, і Постійні (pi = 3.14159…, e = 2,71828…., i = $\sqrt{-1}$)
Приклад.
>> (2 + 3 * pi) / 2*i
ans = 0.0000 + 5.7124i

Вбудовані функції

Основні математичні функції, наявні в MATLAB: abs (#) cos (#) exp (#) log (#) log10 (#) cosh (#) sin (#) tan (#) sqrt (#) floor (#) acos (#) tanh (#).

Наступний приклад ілюструє, як можна комбінувати функції і арифметичні операції. Опис інших функцій можна знайти, використовуючи легко доступну довідку в діалоговому режимі, набравши в командному вікні команду help і ім’я функції.

Приклад. >> 3 * cos (sqrt (4.7))
ans = –1.6869

За замовчуванням виводиться приблизно п’ять десяткових значущих цифр. Команда format long дозволяє вивести приблизно 15 десяткових значущих цифр.

Приклад. >> format long >> 3 * cos (sqrt (4.7))
ans = –1.68686892236893

Оператори присвоювання

Для присвоювання виразам (іменам) різних дій застосовується знак рівності.

Приклад. > a = 3-floor (exp (2.9))
a = –15

Крапка з комою, розташована в кінці виразу, не передбачає виводу результату команди на екран комп’ютера.

Приклад. >> b = sin (a);

```
>> 2 * b ^ 2
ans =
0.8457
```

Оператори відносин

==	Дорівнює
~=	Не дорівнює
<	Менше
>	Більше
<=	Менше або дорівнює
>=	Більше або дорівнює

Логічні оператори

~	Not (доповнення)
&	And (істина, якщо істинні обидва операнди)
|	Or ((істина, якщо один з двох або обидва операнди істинні)

Булеві величини

1	True
0	False

Матриці

Всі змінні в MATLAB інтерпретуються як матриці або масиви. Матриці можна вводити безпосередньо:

Приклад.
```
>> A = [1 2 3; 4 5 6; 7 8 9]
   A = 1 2 3
       4 5 6
       7 8 9
```

Крапка з комою використовується для поділу рядків матриці. Елементи матриці слід розділяти одиночним пробілом. Альтернативний спосіб введення матриці – рядок за рядком.

Приклад.
```
>> A = [1 2 3
       4 5 6
       7 8 9]
   A = 1 2 3
       4 5 6
       7 8 9
```

Матриці можна формувати, використовуючи вбудовані функції.

Приклад.
```
>> Z = zeros (3,5); % Створення нульової матриці розміру 3x5.
>> X = ones (3,5); % Створення матриці розміру 3x5, що
                   % складається з одиниць.
>> Y = 0: 0.5: 2 % Вивід на екран матриці розміру 1x5.
   Y =
   0 0.5000 1.0000 1.5000 2.0000
>> cos (Y) % Створення матриці розміру 1x5, кожен елемент
           % якої є cos від Y
    ans =
```

1.0000 0.8776 0.5403 0.0707 -0.4161

Маніпулювання з елементами матриці.

Приклад. % Виділення одного елемента матриці A
>> A (2,3)
ans =
6

Приклад. % Виділення підматриці A
>> A (1: 2, 2: 3)
ans =
2 3
5 6

Приклад. % Інший спосіб виділення підматриці A
>> A ([1 3], [1 3])
ans =
1 3
7 9

Приклад. % Присвоєння нового значення елементу матриці A
>> A (2,2) = tan (7.8);

Додаткові команди для матриць можна знайти, використовуючи довідку в діалоговому режимі або документацію до пакету прикладних програм.

Операції з матрицями

+ Додавання
- Віднімання
* Добуток
^ Піднесення до степеня
' Спряження та транспонування

Приклад. >> B = [1 2, 3 4],;
>> C = B' % C дорівнює транспонованій матриці B
C =
1 3
2 4
>> 3 * (B * C) ^ 3
ans =
13080 29568
29568 66840

Операції з масивами

Однією з найбільш корисних властивостей пакета MATLAB є можливість виконувати операції над окремими елементами матриці. Така операція була продемонстрована вище, коли елементи матриці розміру 1x5 записувалися як cos від елементів іншої матриці. Операції додавання, віднімання і добутку матриці на скаляр завжди здійснюються поелементно, але це не відноситься до операцій множення, ділення і піднесення матриці до ступеня. Дані три операції можна проводити поелементно, як і попередні, але

з крапкою перед символом операції: **.*, ./ .^.** Важливо зрозуміти, як і коли саме їх можна використовувати. Операції над масивами є вирішальними для ефективної побудови та виконання програм пакетом MATLAB.

Приклад.
```
>> A = [1 2, 3. 4];
>> A ^2 % Добуток матриць A*A або піднесення в квадрат
ans =
7 10 15 22
>> A .^2    % Квадрат кожного елемента A
ans =
1 4
9 16
>> cos (A./2)  % Ділення кожного елемента A на 2 і
                % обчислення cos від кожного елемента
ans =
0.8776 0.5403 0.0707 -0.4161
```

Введення значень з клавіатури і виведення на екран

Для організації діалогового вводу/виводу використовуються функції **input** (введення з клавіатури)і **disp** (виведення значення на екран).

Функція disp призначена для виведення її параметра на екран.

Функція disp має наступний синтаксис:

disp (Значення)

Приклад. >> disp('Hello') % Виведення рядка символів

Функція input має наступний синтаксис:

змінна = **input** (рядок)

При виконанні цієї функції спочатку виводиться рядок, потім відбувається зупинка роботи програми і очікується введення значення. Введення підтверджується натисканням клавіші Enter, після чого введене значення присвоюється змінній.

Приклад.
```
% Введення і обчислення квадрату числа
>> x=('Input number: ')
z=input(x);
y=z.^2;
disp(y)
```

Цикли і умовні оператори

Крім програм з лінійною структурою, інструкції яких виконуються строго по порядку, MATLAB дозволяє створювати програми, структура яких нелінійна. Для створення таких програм застосовуються наступні умовні оператори: **if, for, while**.

Синтаксис оператора **for**:

for (**var** = вираз)
    Послідовність операторів
end

Цикли типу **for ... end** зазвичай використовуються для організації обчислень із заданим числом повторюваних циклів.

Вираз найчастіше записується у вигляді **s: d: e**, де: **var** – ім'я змінної циклу, **s** – початкове значення змінної циклу, **d** – прирощення цієї змінної та **e** – кінцеве значення, при досягненні якого цикл завершується. За замовчуванням **d** дорівнює 1.

Приклад. % Виведення на екран значень **g** з діапазону -1…0, з кроком -0.1

```
>> for g = 1.0: -0.1: 0.0
        disp(g)
    end
```

Синтаксис оператора **while**:

while Умова
    Послідовність операторів
end

Цикл типу while... end виконується до тих пір, поки залишається істинним умова:

Приклад. % Обчислення суми чисел віл 1 до 100

```
n = 1;
nSum = 1;
while n < 100
   n = n + 1;
   nSum = nSum + n;
end
```

Синтаксис оператора **if**:

if Умова
    Послідовність операторів
else
    Послідовність операторів
end

Приклад.

```
x=('Input number: ');
z=input(x);
if z>5
   F=z+2;
else
  F=z^2;
end;
disp(F);
```

Для зупинки програми використовується оператор pause. Він використовується в таких формах:

1) pause – зупиняє обчислення до натискання будь-якої клавіші;
2) pause (N) – зупиняє обчислення на N секунд;
3) pause on – включає режим обробки пауз;
4) pause off – вимикає режим обробки пауз.

М-файли сценаріїв і функцій

М-файли системи MATLAB діляться на два класи:

- файли-сценарії, що не мають вхідних параметрів;
- файли-функції, що мають вхідні параметри.

Файл-сценарій або Script-файл не має списку вхідних параметрів. Він використовує глобальні змінні, тобто такі змінні, значення яких можуть бути змінені в будь-який момент сеансу роботи і в будь-якому місці програми. Для запуску файлу-сценарію з командного рядка MATLAB достатньо вказати його ім'я в цьому рядку.

Файл-сценарій має наступну структуру:

% Основний коментар – один рядок (обов'язковий)

% Додатковий коментар – будь-яке число рядків (не обов'язковий)

Тіло файлу – будь-які вирази, команди і оператори.

Файл-функція відрізняється від файлу-сценарію насамперед тим, що створена ним функція має вхідні параметри, список яких вказується у круглих дужках. Використовувані в файлах-функціях змінні і імена параметрів є локальними змінними, зміна значень яких в тілі функції не впливає на значення змінних з таким ж іменами за межами функції.

Файл-функція має наступну структуру:

function var = ***ім'я m-файлу*** **(Список параметрів переданих значень)**

% Основний коментар – один рядок (обов'язковий)

% Додатковий коментар – будь-яке число рядків (необов'язковий)

Тіло файлу – будь-які вирази, команди і оператори.

var = вираз.

Остання інструкція тіла "var = вираз" вводиться, якщо потрібно, щоб функція повертала результат обчислень. Якщо необхідна більша кількість вихідних параметрів, структура модуля буде мати наступний вигляд:

function [var1, var2, ...] = ***ім'я m-файлу*** **(Список параметрів переданих значень)**

% Основний коментар – один рядок (обов'язковий)

% Додатковий коментар – будь-яке число рядків (необов'язковий)

Тіло файлу – будь-які вирази, команди і оператори.

var1 = вираз

var2 = вираз

Імена **var, varl, var2**, ... для значень, що повертаються є глобальними або відомими в тілі програми, яка викликає М-функції.

Приклад. Визначимо функцію fun (x) = 1 + x-x^2 / 4 в М-файлі fun.m.

В редакторі вона записується в такий спосіб.

```
function y = fun(x)
% Example M-script
%   Calculate y=1+x-x^2/4
y=1+x-x.^2./4;
end
```

Коментар з М-файлу можна вивести на екран за допомогою команди **help** ***ім'я m-файлу***, причому вивід проводиться до першого порожнього рядка.

Для позначення змінних можна вживати різні літери і для назви функцій – різні імена, але слід використовувати один і той же формат. Функцію, одного разу записану як М-файл з ім'ям **fun.m**, можна викликати в MATLAB Command Window так само, як будь-яку іншу функцію.

Приклад. >> p=1

```
>> fun(p)
 ans=
     1.7500
```

Приклад.
```
>> cos (fun (3))
ans =
  -0.1782
```

Корисним і ефективним способом обчислення функцій є використання команди feval. Вона вимагає, щоб функція викликалася, як рядок.

Приклад.
```
>> feval ('fun', 4)
 ans =
 1
```

Перелік рекомендованої літератури

1. Дьяконов В. П. MATLAB. Полный самоучитель. – М.: ДМК Пресс, 2012. – 768 с.: ил.

2. Лазарев Ю. Моделирование процессов и систем в MATLAB. Учебный курс. – СПб.: Питер; Киев: Издательская группа BHV, 2005. – 512 с.: ил.

Практичне заняття 4

Тема: Побудова функцій належності в MATLAB

Мета: вивчення основних понять теорії нечітких множин та теорії нечіткого логічного виведення; вивчення основних видів функцій належності нечітких множин; ознайомлення зі складом і можливостями інструментарію нечіткої логіки Fuzzy Logic Toolbox пакету MATLAB; набуття практичних навичок роботи в пакеті Fuzzy Logic Toolbox.

Завдання

1. Побудувати за допомогою операторів і функцій MATLAB графіки трикутної, трапецієподібної і гаусових функції належності згідно з варіантом (таблиця 1).
2. Функції створити в редакторі MATLAB і зберегти у виді файлу (*.m). Приклади наведені у п.3. додатка А.
3. Побудувати ці функції в редакторі FIS.
4. Проаналізувати отримані результати і зробити висновки.

Таблиця 1 – Варіанти функцій належності

№	Діапазон (x1…x2)	Трикутна			Трапецієподібна				Гаусова	
		a	b	c	a	b	c	d	c	s
1.	0… 1	0,15	0,45	0.84	0,12	0,32	0,61	0.81	0,5	1, 05, 025
2.	5…10	5,3	6,45	9.8	5,1	6,9	7,6	9,1	7,5	1, 05, 025
3.	-10…- 4	-9,5	-7,2	-4,7	-9,1	-7,8	-6,0	-4,2	-7,0	1, 05, 025
4.	-2…5	-1,81	0	4,8	-1,9	0	2,6	4,5	-7,5	1, 05, 025
5.	15…30	16	24	29	15,1	19,3	26	28,7	22,5	1, 05, 025
6.	-2…2	-1,8	0,2	1,83	-1,7	-1,2	0,8	1,6	0	1, 05, 025
7.	2…8	2,1	4,5	7.8	2,3	3,1	5,3	7,9	5	1, 05, 025
8.	45…60	45,1	51,3	57,8	47	50,1	54,1	59	52,5	1, 05, 025
9.	-6…0	-5,9	-3,2	-0,12	-5,6	-4,7	-2,3	-0,45	-3	1, 05, 025
10.	14…23	14,6	19,2	22,6	0,1	15,1	17,3	20,7	22,5	1, 05, 025
11.	0…51	3,0	27	48	4	11	27	35	25,5	1, 05, 025
12.	-20…0	-19,1	-9,1	-0.8	0,1	0,3	0,6	0,9	-10	1, 05, 025
13.	-56…4	-51	-15	2	-47	-30	-20	-3	-26	1, 05, 025
14.	10…25	11	14	21	12	15	18	23	17,5	1, 05, 025
15.	-2… 14	-1,0	6	11	-1,1	0	9	13,5	6	1, 05, 025
16.	10…34	10,3	21	31	11	14	24	31	17	1, 05, 025
17.	-3…0	-2,7	-1,5	-0.8	-2,9	-1,7	-1	-0,1	-1,1	1, 05, 025
18.	21…64	27	41	59	28	33	39	61	42,5	1, 05, 025
19.	-10…-1	-8,3	-5,2	-2,1	-9,1	-6,2	-5,0	-2,0	-5,5	1, 05, 025
20.	11…24	11,3	16	21	12	14	18	20	17,5	1, 05, 025
21.	0…1	0,08	0,57	0.81	0,1	0,3	0,6	0,7	0,9	1, 05, 025
22.	-1…0	-0,9	-0,56	-0.12	-0,91	-0,8	-0,7	-0,3	-0,2	1, 05, 025
23.	-5…15	0,1	0,5	0.8	0,1	0,3	0,6	0,9	5,0	1, 05, 025
24.	22…30	22,5	27,3	28,5	22,5	24,0	26,0	28,6	26	1, 05, 025
25.	-4…14	-3,1	1,1	11	-3,0	0	6	10	12	1, 05, 025
26.	-6…10	-5	0	7.2	-4,1	-1	3,2	6,3	2,0	1, 05, 025
27.	2…14	2,1	5,8	10,3	2,7	3,7	8,1	11,1	8,0	1, 05, 025
28.	4…10	4,1	6,1	8,9	4,5	5,5	6,5	9,0	7,0	1, 05, 025
29.	5…15	6,1	7,9	12	7,2	9,0	11,2	14	10	1, 05, 025
30.	-3…0	-2,7	-1,5	-0.8	-2,5	-2,0	-1,0	-0,4	-1,5	1, 05, 025

Зміст звіту

1. Титульна сторінка.
2. Тема практичного заняття.
3. Мета заняття.
4. Завдання.
5. Опис виконання завдань по пунктам з наданням рисунків і скріншотів.
6. Текст програми MATLAB
7. Висновки.

Контрольні питання.

1. Поняття нечіткої множини.
2. Операції над нечіткими множинами.
3. Поняття лінгвістичної змінної.
4. Типи функцій належності в MATLAB.

Теоретичні відомості

1. Визначення нечіткої множини

Нечіткою множиною A на універсальній множині X називається сукупність пар $(\mu_A(x), x)$, де $\mu_A(x)$ – ступінь належності елемента $x \in X$ нечіткій множині A. Ступінь належності – це число з діапазону [0, 1]. Чим вище ступінь належності, тим більшою мірою елемент універсальної множини відповідає властивостям нечіткої множини.

Функцією належності (ФН) називається функція, що обчислює ступінь належності елемента універсальної множини до нечіткої множини.

У разі безперервної множини U використовують таке позначення

$$\tilde{A} = \int_{x \in X} \frac{\mu_A(x)}{x}$$

Знак $\int$ позначає безперервну сукупність величин, а не операцію інтегрування. Якщо універсальна множина є скінченною $U = \{u_1, u_2, ..., u_k\}$, тоді нечітка множина A записується як:

$$\tilde{A} = \sum_{i=1}^{k} \frac{\mu_A(x_i)}{x_i} \quad \text{або} \quad \tilde{A} = \left(\frac{\mu_A(u_1)}{u_1}, \frac{\mu_A(u_2)}{u_2}, ..., \frac{\mu_A(u_k)}{u_k} \right)$$

де Σ позначає дискретну операцію.

2. Способи формування функцій належності в середовищі MATLAB

Інструментарій нечіткої логіки в складі пакету MATLAB містить 11 вбудованих типів функцій належності на основі кусково-лінійних функцій, розподілу Гауса, сигмоїдної кривої, квадратичних і кубічних поліноміальних кривих. До найбільш простих ФН можна віднести трикутну і трапецієподібну.

Трикутні ФН.

В аналітичному вигляді трикутна ФП може бути задана наступним чином:

$$f(x,a,b,c)=\begin{cases}0, x<a\\ \dfrac{x-a}{b-a}, a\le x\le b,\\ \dfrac{c-x}{c-x}, b\le x\le c,\\ 0, x>c\end{cases}$$

Однією з основних переваг розглянутих вище функцій є їх простота, реалізація даних функцій у системі MATLAB зображена на рис. 1.

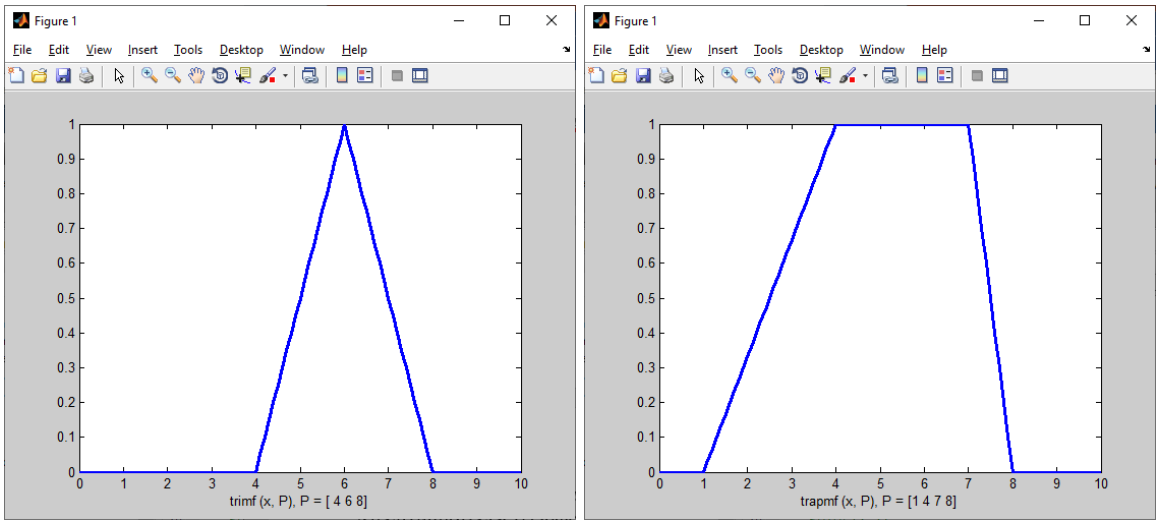


Рисунок 1 – Трикутна і трапецієвидна функції належності

Гаусові функції належності.

Аналітично задається формулою:

$$f(x,\sigma,c)=e^{\frac{-(x-c)^2}{2\sigma^2}}$$

На основі функції розподілу Гауса можна побудувати ФН двох видів: просту функцію належності Гауса і двосторонню, утворену за допомогою різних функцій розподілу Гауса (рис.2).

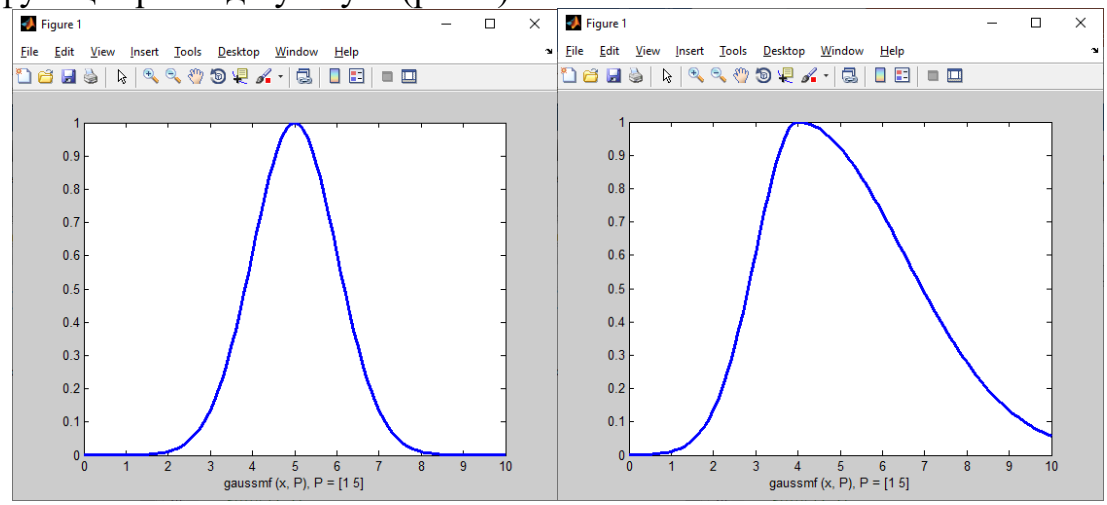


Рисунок 2 – Проста і двостороння гаусові функції належності

Функція *gaussmf* задає функцію належності у вигляді симетричною гаусовської кривої.

Параметри функції *gaussmf* геометрично інтерпретуються наступним чином:

c – координата максимуму функції належності;

σ – коефіцієнт концентрації функції належності.

Функція *gaussmf* застосовується для завдання гладких симетричних функцій належності. Синтаксис функції:

y = gaussmf(x, params).

Функція *gaussmf* має два вхідних аргументи:

x – вектор, для координат якого необхідно розрахувати ступінь належності; *params* – вектор параметрів функції належності. Порядок завдання параметрів – *[σ ,c].*

Функція повертає вихідний аргумент у вигляді масиву, що містить ступені належності координат вектору.

Двостороння гаусова функція належності задається виразом

y = gauss2mf(x, params).

Ця функція належності складається з комбінації двох гаусових функцій належності:

Порядок завдання параметрів – *[σ_1 ,c_1, σ_2 ,c_2].*

якщо $c_1 < c_2$, то

$$\mu(x) = \begin{cases} \exp\left(\dfrac{(x-c_1)^2}{-2a_1^2}\right), & x < c_1 \\ 1, & c_1 \le x \le c_2 \\ \exp\left(\dfrac{(x-c_2)^2}{-2a_2^2}\right), & x > c_2 \end{cases}$$

якщо $c_1 > c_2$, то

$$\mu(x) = \begin{cases} \exp\left(\dfrac{(x-c_1)^2}{-2a_1^2}\right), & x < c_2 \\ 1, & c_2 \le x \le c_1 \\ \exp\left(\dfrac{(x-c_2)^2}{-2a_2^2}\right), & x > c_1 \end{cases}$$

3. Побудови функцій належності в MATLAB

Опис трикутної функції належності в MATLAB (функція *trimf*):

y = trimf(x, [a b c]),

де вектор *x* – базова множина, на якій визначається ФН. Величини *a* і *c* задають основу трикутника, *b* – його вершину.

Приклад використання трикутної ФН в системі MATLAB:

```
x = 0: 0.1: 10;          % Завдання базової множини з кроком 0.1
y = trimf (x, [4 6 8]); % Обчислюється трикутна ФН для вершин (a = 4, b = 6, c =8)
plot (x, y);             % Виводиться графік обчисленої функції
title ('Membership function'); % Вивід заголовка графіка
xlabel ('Triangular membership function [ 4 6 8]');  % Підпис під віссю x
```

Приклад побудови трапецієподібної і гаусової ФН на одному графіку.

```
x = 0: 0.1: 10;                 % Завдання базової множини
y = trapmf (x, [1 4 7 8]); % Обчислюється трапецієподібна ФН для
                                вершин
                                %(a = 1, b = 4, c =7 d = 8)
plot (x, y);                    % Виводиться графік функції
title ('Membership functions'); % Вивід заголовка графіка
hold on;                        % Тримати графік активним
y1 = gaussmf (x, [1 5]);  % Обчислюється гаусова ФН (σ = 1 ,c = 5)
plot (x, y1);                   % Виводиться графік гаусової ФН
xlabel ('Trapezius+ Gaussian membership function'); % Підпис під віссю x
                                                    %графіка
hold off;                       % Відключити режим активності графіка
```

Приклад побудови трьох гаусових ФН на одному графіку різними кольорами і типами кривих.

```
x = 0: 0.1: 10;                 % Завдання базової множини
y = gaussmf (x, [1 5]);         % Обчислюється перша гаусова ФН
hold on;
plot (x, y, ['+', 'k']);        % Виводиться графік функції y точками у
                                %виді символу '+' чорного кольору.
title ('Membership functions'); % Вивід заголовка графіка
xlabel ('Gaussian membership functions'); % Підписується графік під
                                          % віссю абсцис
y1 = gaussmf (x, [0.5 5]);      % Обчислюється друга гаусова ФН
plot (x, y1, 'r');              % Виводиться графік функції y1
                                % червоного кольору;
y2 = gaussmf (x, [0.25 5]);     % Обчислюється третя гаусова ФН
plot (x, y2, '-.');             % Виводиться графік функції y2
                                % штрих-пунктиром;
hold off;
```

4. Загальні відомості про побудову графіків в MATLAB

Команда plot (y) будує графік елементів одновимірного масиву y в залежності від номера елемента; якщо елементи масиву y комплексні, то будується графік plot (real (y), imag (y)). Якщо Y – двовимірний дійсний масив, то будуються графіки для стовпців; в разі комплексних елементів їх уявні частини ігноруються.

Команда plot (x, y) відповідає побудові звичайної функції, коли одновимірний масив x відповідає значенням аргументу, а одновимірний масив y – значенням функції. Коли один з масивів X або Y або обидва двовимірні, реалізуються наступні побудови:

• якщо масив Y двовимірний, а масив x одновимірний, то будуються графіки для стовпців масиву Y в залежності від елементів вектора x;

• якщо двовимірним є масив X, а масив y одновимірний, то будуються графіки стовпців масиву X в залежності від елементів вектора y;

• якщо обидва масиви X і Y двовимірні, то будуються залежності стовпців масиву Y від стовпців масиву X.

Команда *plot (x, y, s)* дозволяє виділити графік функції, вказавши спосіб відображення лінії, спосіб відображення точок, колір ліній і точок за допомогою рядкової змінної s, яка може включати до трьох символів з табл. 2:

Таблиця 2 – Значення змінної *s* для способів відображення графіків

Лінія		Точка		Колір	
Тип	Символ	Тип	Символ	Тип	Символ
Безперервна	‘-‘	Крапка	‘.’	Жовтий	‘y’
Штрихова	‘--‘	Плюс	‘+’	Фіолетовий	‘m’
Подвійний пунктир	‘:’	Зірочка	‘*’	Блакитний	‘c’
Штрих-пунктирна	‘-.’	Кільце	‘o’	Червоний	‘r’
		Хрестик	‘x’	Зелений	‘g’
		Ромб	‘d’	Синій	‘b’
		Квадрат	‘s’	Білий	‘w’
		П’ятикутник	‘p’	Чорний	‘k’
		Шестикутник	‘h’		
		Трикутник (уверх)	‘a’		

Графіки виводяться в окремих графічних вікнах за допомогою команди виду figure (n), де n – номер графічного вікна. На одному графіку можна побудувати кілька кривих, що відрізняються кольором і типами ліній і точок. Графіки можуть бути скопійовані і вставлені в інші додатки: Word, Excel, PowerPoint та ін. Для цього використовується команда Edit / Copy / Figure вікна графіки.

Часто використовувані команди при побудові графіків

plot (t, y) % Графік неперервної функції *y (t)*
plot (x1, y1, x2, y2) % Графіки залежностей *y1* від *x1* і *y2* від *x2*
stem (x, y) % Графік дискретної функції (сигналу) *y(x)*
stairs (x, y) % Графік у вигляді ступінчастою лінії
loglog (x, y) % Графік з логарифмічними масштабами по *x* і *y*
semilogx (x, y) % Логарифмічний масштаб по *x* і лінійний по *y*
polar (phi, r) % Графік в полярних координатах
title ('назва') % Вивід заголовка графіка
xlabel ('час') % Підпис під віссю *x*
ylabel ('Напруга') % Підпис осі *y*
legend ('АЧХ системи') % Вивід написів для пояснення графіків
axis ([xmin, xmax, ymin, ymax]) % Установка масштабів по осям *x* і *y*
xlim ([xmin, xmax]) % Установка масштабу по осі *x*
ylim ([ymin, ymax]) % Установка масштабу по осі *y*
figure (n) % Встановлює фігуру (вікно) *n* активним
subplot (r, c, n) % Розбиває графічне вікно на *r* * *c* підвікон і %*subplot (r,c,n)* встановлює підвікно *n* активним.
grid on % до графіку додається сітка
hold on % дозволяє побудувати кілька графіків у вікні

```
hold off                % відміняє hold on для поточного графіка
text                    % дозволяє розмістити текст на графіку
zoom on/off             % включення/вимикання збільшення
                        % фрагментів графіка з використанням
                        % лівої і правої кнопок миші
```

Перелік рекомендованої літератури

1. Штовба С.Д. Проектирование нечетких систем средствами MATLAB. – М: Горячая линия – Телеком, 2007, - 288с.: ил.
2. Дьяконов В. П. MATLAB. Полный самоучитель. – М.: ДМК Пресс, 2012. – 768 с.: ил.
3. Леоненков А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ – Петербург, 2005. – 736 с.: ил.
4. Рутковская Д., Пилинский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Ж Пер. с польск. И.Д. Рудинского. М.: Горячая линия – Телеком, 2006. – 452с.: ил.
5. Пегат А. Нечеткое моделирование и управление; пер. с англ. / А. Пегат. – М. : БИНОМ, 2009. – 798 с.

Практичне заняття 5

Тема: Створення системи нечіткого керування засобами MATLAB

Мета: вивчення основ створення систем нечіткого керування об'єктами, набуття практичних навичок роботи в пакеті Fuzzy Logic Toolbox.

Завдання

Для предметної області згідно з варіантом (таблиця 1):

1. Зробити короткий опис предметної області, визначити вхідні і вихідні змінні;
2. Сформулювати мету функціонування системи керування;
3. Визначити діапазони значень вхідних і вихідних змінних;
4. Записати правила функціонування системи.
5. Засобами Fuzzy Logic Toolbox створити систему нечіткого виведення (функції належності вхідних і вихідних змінних, базу нечітких правил).
6. Результати зберегти у виді файлу MATLAB (*.fis).
7. Дослідити створену систему нечіткого виведення на різних значеннях вхідних даних.
8. Проаналізувати отримані результати і зробити висновки. ·

Таблиця 1 – Перелік рекомендованих тем для створення систем.

№ з/п	Тема	Вхід / Кількість термів	Вихід / Кількість термів
1.	Керування душем	Температура води / 3	Кут повороту крану / 3
2.	Керування електр. двигуном	Швидкість обертання / 5	Напруга / 5
3.	Керування системою опалення.	Температура / 3 Вік людини / 5	Клапан / 3
4.	Визначення рангу студента	Середня успішність / 3 Термін виконання / 3	Ранг / 3
5.	Визначення суми кредиту	Кількість наданих кредитів / 3 Розмір попередніх кредитів / 3	Сума кредиту / 3
6.	Керування клапаном для води	Рівень води / 3	Клапан / 3
7.	Керування кермом автомобіля	Відстань / 3 Положення перешкоди відносно напряму руху / 3	Кут повороту керма / 5
8.	Конкурента здатність	Кількість втрачених клієнтів / 3 Кількість нових клієнтів / 3	Прибуток / 3
9.	Керування опаленням	Температура / 3	Кут повороту крану / 3
10.	Керування кондиціонером	Температура / 3	Напруга / 3

Номер варіанту для таблиці 1 визначається як залишок від ділення номеру із журналу на кількість варіантів,
наприклад, 23/10 – залишок 3.

Зміст звіту

1. Титульна сторінка.
2. Тема практичного заняття.
3. Мета заняття.
4. Номер варіанту та завдання до роботи.
5. Опис виконання завдань по пунктам з наданням рисунків і скріншотів.
6. Текст створеного *.fis файлу.
7. Висновки, що відображують результати виконання роботи та їх критичний аналіз.

Контрольні питання

1. Структура нечіткого логічного виведення
2. Поняття лінгвістичної змінної
3. Метод нечіткого виведення Мамдані.
4. Методи дефазіфікації для нечіткого виведення Мамдані.
5. Методи дефазіфікації для нечіткого виведення Сугено.

Теоретичні відомості

Операції з нечіткою логікою у пакеті MATLAB дозволяє виконувати модуль *Fuzzy Logic Toolbox*. Він дозволяє створювати системи нечіткого логічного виведення і нечіткої класифікації в рамках середовища MATLAB, з можливістю інтегрування в Simulink.

Базовим поняттям Fuzzy Logic Toolbox є *FIS-структура* – система нечіткого виведення (Fuzzy Inference System). FIS-структура містить усі необхідні дані для реалізації функціонального відображення “входи-виходи” на основі нечіткого логічного виведення відповідно до схеми, приведеної на рис. 1.

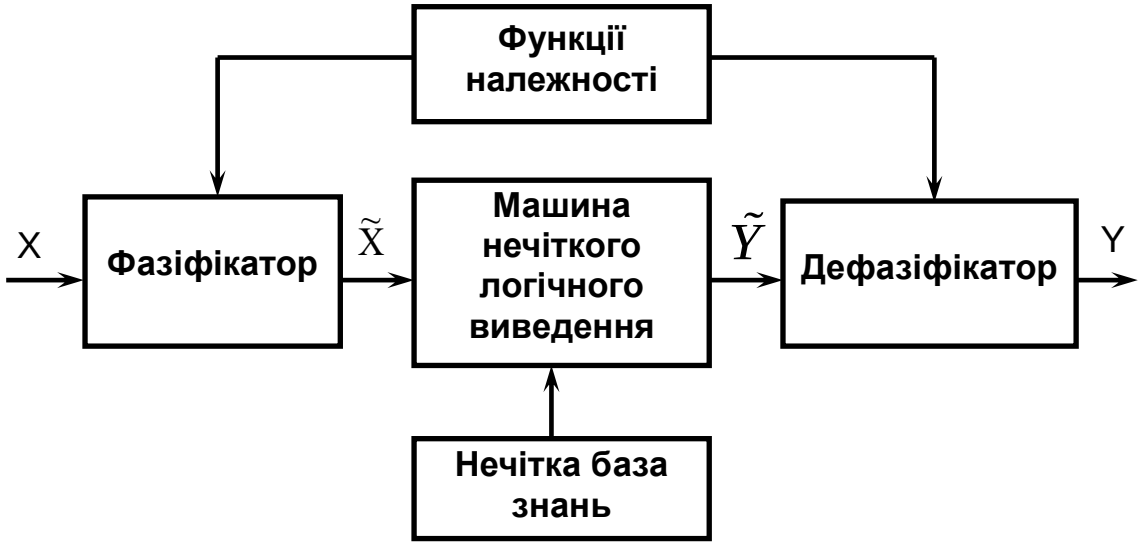


Рисунок 1 – Нечітке логічне виведення

Позначення: X – вхідний чіткий вектор; $\widetilde{X}$ – вектор нечітких множин, що відповідає вхідному вектору X; $\widetilde{Y}$ – результат логічного виведення у виді вектору нечітких множин; Y – вихідний чіткий вектор.

Припустимо, що базу знань утворюють два нечітких правила:

Правило 1: Якщо $x_1 \in A_1^1$ і $x_2 \in A_1^2$, То $y \in B^1$;

Правило 2: Якщо $x_1 \in A_2^1$ і $x_2 \in A_2^2$, То $y \in B^2$,

де:

x_1, x_2 – імена вхідних змінних,

y – ім'я вихідної змінної,

A_1^1, A_1^2, A_2^1, A_2^2 – функції належності вхідних змінних,

B^1, B^2 – функції належності вихідних змінних.

Алгоритм Мамдані

На рис. 2 представлений порядок дій алгоритму

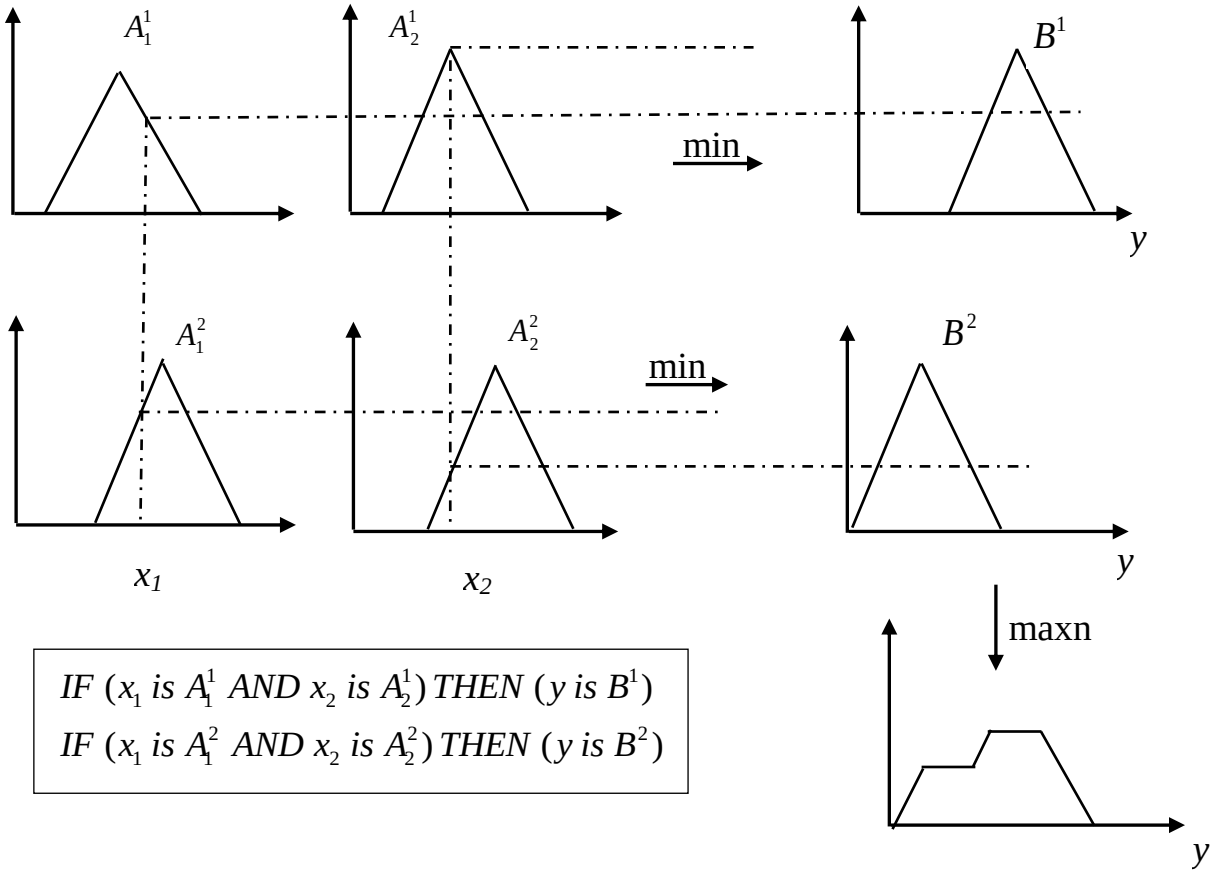


Рисунок 2 – Порядок дій алгоритму Мамдані

Алгоритм Мамдані:

1. Введення нечіткості (fuzzification): для заданих (чітких) значень аргументів $x_1 = x_1(0)$, $x_2 = x_2(0)$ знаходяться степені істинності для передумов правил П1, П2.

2. Знаходиться рівні відсічення для передумов кожного з правил (з використання правила мінімуму):

$\alpha_1 = A_1^1 [x_1(0)] \wedge A_1^2 [x_2(0)]$

$\alpha_2 = A_2^1 [x_1(0)] \wedge A_2^2 [x_2(0)]$,

де $\wedge$ – операція нечіткого логічного мінімуму

3. Знаходиться усічені рівні вихідних функцій належності

$B^1 [y(0)] = (\alpha_1 \wedge B^1)$

$B^2 [y(0)] = (\alpha_2 \wedge B^2)$

4. З використанням операції max (позначеної як "V") проводиться об'єднання знайдених усічених функцій, що призводить до отримання підсумкової нечіткої множини для змінної виходу з функцією належності:

$B = (\alpha_1 \wedge B^1) \vee (\alpha_2 \wedge B^2)$

5. Приведення до чіткості (для знаходження $y(0)$) проводиться, наприклад, методом центроїда (як центр ваги фігури під кривою функції належності)

Fuzzy Logic Toolbox містить наступні категорії програмних інструментів: функції; інтерактивні модулі з графічним користувальницьким інтерфейсом (GUI); блоки для пакета Simulink; демонстраційні приклади.

FIS Editor

FIS Editor (FIS редактор) призначений для створення, збереження, завантаження і графічного друкованого представлення результатів нечіткого логічного виведення, а також для редагування: типу системи; найменування системи; кількості вхідних і вихідних змінних; найменування вхідних і вихідних змінних; параметрів нечіткого логічного виведення (рис. 3).

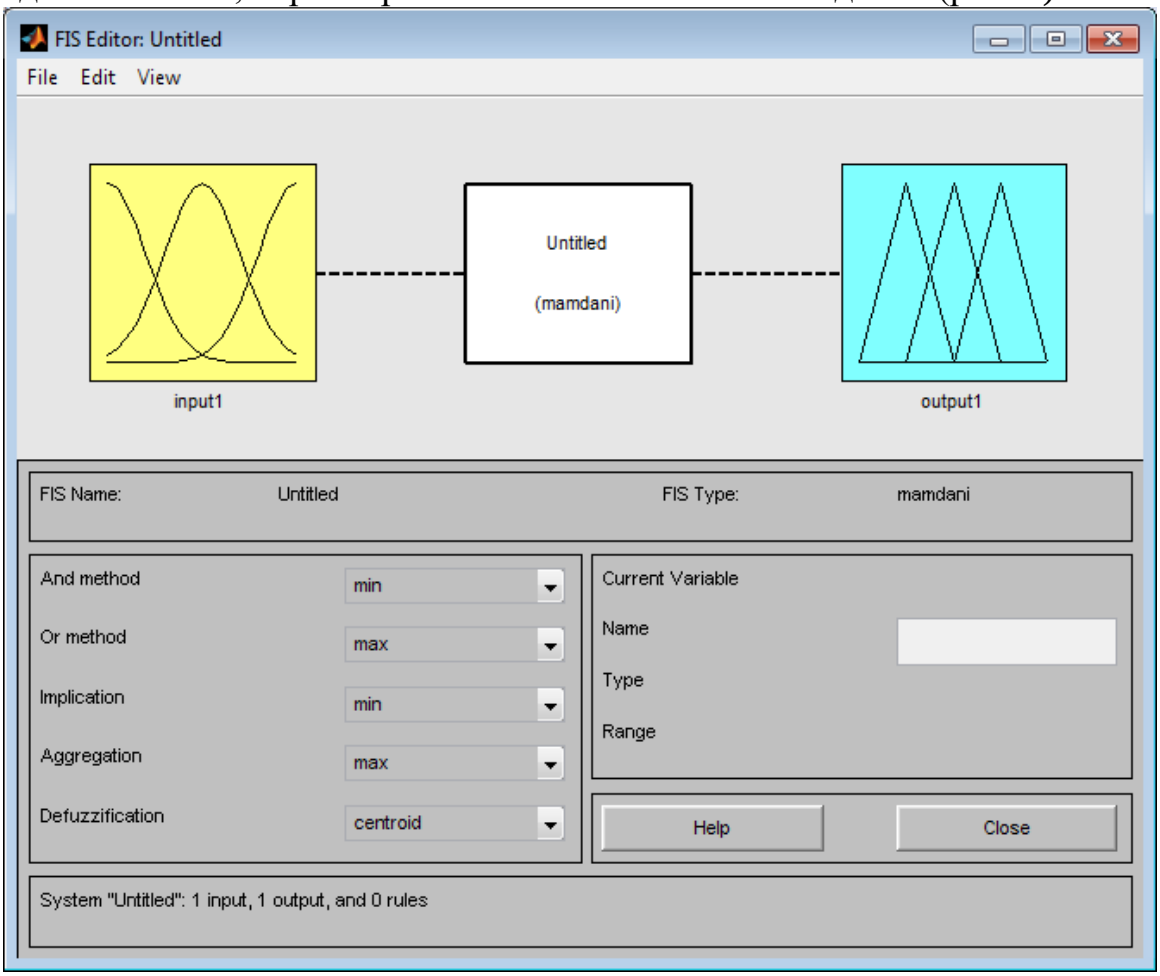


Рисунок 3 – Робоче вікно FIS Editor

Завантаження FIS Editor відбувається за допомогою команди fuzzy в робочій області MATLAB. У результаті з'являється інтерактивне графічне вікно. У нижній частині графічного вікна FIS Editor розташовані кнопки **Help** і **Close** для виклику вікна довідки і закриття редактора.

FIS Editor містить 8 меню. Це три загальносистемних меню – **File, Edit, View,** і п'ять меню для вибору параметрів нечіткого логічного виведення – **And Method, Or Method, Implication, Aggregation і Defuzzification**.

Меню File

Це загальне меню для всіх GUI-модулів використовуваних із системами нечіткого логічного виведення.

Команди меню **File.**

New FIS – створення нової систему нечіткого логічного виведення. При виборі цієї команди з'являться дві альтернативи: Mamdani (Ctrl+N) і Sugeno, що визначають тип створюваної системи. За промовчанням створюється типу Mamdani.

Import – завантаження раніше створеної систему нечіткого логічного виведення. При виборі команди з'являться дві альтернативи **From Workspace...** і **From file…**, що дозволяють завантажити систему нечіткого логічного виведення з робочої області MATLAB і з диска, відповідно. При виборі команди **From Workspace...** з'явиться діалогове вікно, у якому необхідно вказати ідентифікатор системи нечіткого логічного виведення, що знаходиться в робочій області MATLAB. При виборі команди **From file…** з'явиться діалогове вікно, у якому необхідно вказати ім'я файлу системи нечіткого логічного виведення. Файли систем нечіткого логічного виведення мають розширення **.fis** . Завантажити систему нечіткого логічного виведення з файлу можна також натисканням Ctrl+N чи командою fuzzy FIS_name, де FIS_name – ім'я файлу системи нечіткого логічного виведення.

Export – копіювання систему нечіткого логічного виведенняю При виборі команди з'являться дві альтернативи **To Workspace...** і **To file…**, що дозволяють скопіювати систему нечіткого логічного виведення в робочу область MATLAB і на диск, відповідно. При виборі команди **To Workspace...** (Ctrl+T) з'явиться діалогове вікно, у якому необхідно вказати ідентифікатор системи нечіткого логічного виведення, під яким вона буде збережена в робочій області MATLAB. При виборі команди **To file… (**Ctrl+S**)** з'явиться діалогове вікно, у якому необхідно вказати ім'я файлу системи нечіткого логічного виведення.

Print (Ctrl+P) **–** друк копії графічного вікна.

Close (Ctrl+W) **–** закриття графічного вікна.

Меню Edit

Undo (Ctrl+Z) – скасовує раніше зроблену дію.

Add Variable... – додавання в систему нечіткого логічного виведення ще однієї змінної. При виборі цієї команди з'являться дві альтернативи **Input** і **Output**, що дозволяють додати вхідну і вихідну змінну, відповідно.

Remove Selected Variable (Ctrl+X) – видаляє поточну змінну із системи.

Ознакою поточної змінної є червона окантовка її прямокутника. Призначення поточної змінної відбувається за допомогою однократного щиглика лівої кнопки миші по її прямокутнику.

Membership Function... (Ctrl+2) – відкриває редактор функцій належності.

Rules… (Ctrl+3) – відкриває редактор бази правил (знань). Ця команда може бути також виконана натисканням Ctrl+3.

Меню View

Це загальне меню для всіх GUI-модулів, використовуваних із системами нечіткого логічного виведення. Дане меню дозволяє відкрити вікно візуалізації нечіткого логічного виведення. Команди:

Rules (Ctrl+5) – відкриття списку правил системи.

Surface (Ctrl+6) – вікно виведення поверхні “вхід-вихід”, що відповідає системі нечіткого логічного виведення (команда.

Меню And Method

Це меню встановлює реалізації логічної операції «І». Команди:

min – мінімум;

prod – множення.

Custom… – власна реалізація операції «І». Після натиснення необхідно в графічному вікні, що з’явилося, надрукувати ім’я функції, що реалізує цю операцію. Перед цим функцію треба створити як .m файл.

Меню Or Method

Це меню встановлює реалізації логічної операції "АБО". Команди:

max – множення;

probor – імовірнісне «АБО».

Custom… – власна реалізація логічної операції «АБО». Після натиснення необхідно в графічному вікні, що з’явилося, надрукувати ім’я функції, що реалізує цю операцію. Перед цим функцію треба створити як .m файл.

Меню Implication

Це меню встановлює реалізації логічної операції імплікації. Команди:

min – мінімум;

prod – множення.

Custom… – власна реалізація логічної операції імплікації. Після натиснення необхідно в графічному вікні, що з’явилося, надрукувати ім’я функції, що реалізує цю операцію. Перед цим функцію треба створити як .m файл.

Меню Aggregation

Це меню встановлює реалізації операції об’єднання функцій належності вихідної змінної.

max – максимум;

sum – сума;

probor – імовірнісне "АБО";

Custom… – власна реалізація операції об'єднання. Після натиснення необхідно в графічному вікні, що з'явилося, надрукувати ім'я функції, що реалізує цю операцію. Перед цим функцію треба створити як .m файл.

Меню Defuzzification

Це меню дозволяє вибрати метод дефазифікації. Для систем типу Мамдані запрограмовані наступні методи: centroid – центр ваги; bisector – медіана; lom – найбільший з максимумів; som – найменший з максимумів; mom – середнє з максимумів.

Для систем типу Сугено запрограмовані наступні методи: wtaver – зважене середнє; wtsum – зважена сума.

Користувач також має можливість установити власний метод дефазифікації. Для цього необхідно вибрати команду **Custom…** і в графічному вікні, що з'явилося, надрукувати ім'я функції, що реалізує цю операцію. Перед цим функцію треба створити як .m файл.

Membership Function Editor

Membership Function Editor (Редактор функцій належності) призначений для встановлення інформації про терми-множини вхідних і вихідних змінних, яка включає: кількість термів; найменування термів; тип і параметри функцій належності, у вигляді нечітких множин. Може бути викликаний з будь-якого GUI-модуля, використовуваного із системами нечіткого логічного виведення, командою **Membership Functions...** (Ctrl+2) меню **Edit**.

У FIS-редакторі відкрити редактор функцій належності можна також подвійним щигликом лівою кнопкою миші по області вхідної або вихідної змінних. Загальний вид редактора з указівкою функціонального призначення основних областей графічного вікна наведений на рис. 4.

У нижній частині графічного вікна розташовані кнопки Help і Close, що дозволяють викликати вікно довідки і закрити редактор, відповідно.

Редактор функцій належності містить чотири меню – **File, Edit, View, Type** і чотири вікна введення інформації – **Range, Display Range, Name і Params**. Ці чотири вікна призначені для завдання діапазону зміни поточної змінної, діапазону виведення функцій належності, найменування поточного лінгвістичного терму і параметрів його функції належності, відповідно.

Параметри функції належності можна підбирати й у графічному режимі, шляхом зміни форми функції належності за допомогою технології "Drug and drop". Для цього необхідно позиціонувати курсор миші на знаку режиму "Drug and drop", натиснути на ліву кнопку миші і не відпускаючи її змінювати форму функції належності. Параметри функції належності будуть перераховуватися автоматично.

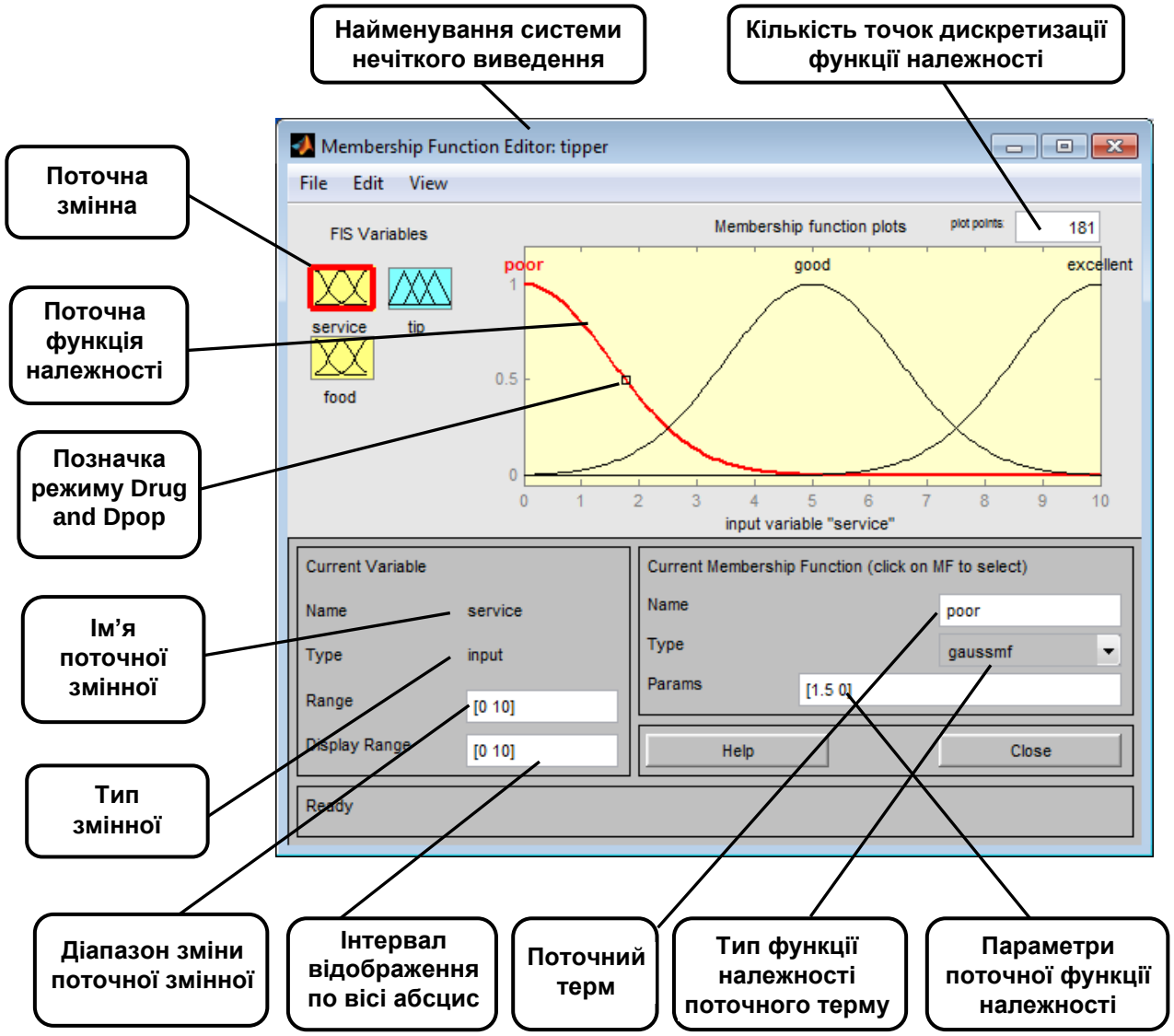


Рисунок 4 – Редактор функцій належності

Меню Edit

Undo (Ctrl+Z) – скасовує раніше зроблену дію.

Add MFs… – додавання термів в терм-множину поточної змінної для лінгвістичної оцінки поточної змінної. При виборі цієї команди з'явиться діалогове вікно, у якому необхідно вибрати тип функції належності і кількість термів. Значення параметрів функцій належності будуть встановлені автоматично таким чином, щоб рівномірно покрити область визначення змінної, заданої у вікні Range. При зміні області визначення у вікні Range параметри функцій належності будуть промаштабовані.

Add Custom MF… – додавання одного лінгвістичного терму, функція належності якого відрізняється від вбудованих. Після вибору цієї команди з'явиться графічне вікно, у якому необхідно надрукувати лінгвістичний терм (поле MF name), ім'я функції належності (поле M-File function name) і параметри функції належності (поле Parameter list).

Remove Selected MF – видаляє поточний терм із терм-множини поточної змінної. Ознакою поточної змінної є червона окантовка її прямокутника. Ознакою поточного терму є червоний колір його функції належності. Для вибору поточного терму необхідно провести позиціонування

курсору миші на графіку функції належності і зробити щиглик лівою кнопкою миші.

Remove All MFs – видаляє всі терми з терм-множини поточної змінної.

FIS Properties… (Ctrl+1) – відкриває FIS-редактор.

Rules… (Ctrl+3) відкриває редактор бази правил (знань).

Вікно Range

Встановлення діапазону зміни лінгвістичної змінної.

Вікно Display Range

Встановлення діапазону відображення лінгвістичної змінної в робочій області вікна.

Вікно Name

Присвоєння терму імені.

Меню Type

Вибір типу функції належності терму поточної змінної: trimf, trapmf, gbellmf, gaussmf, gauss2mf, sigmf, dsigmf, psigmf, pimf, smf, zmf.

Вікно Params

Встановлення параметрів функції належності терму поточної змінної. Наприклад значення трьох точок трикутної функції належності trimf.

Rule Editor (Редактор бази знань)

Rule Editor (*Редактор бази знань)* призначений для формування і модифікації нечітких правил. Редактор бази знань може бути викликаний з будь-якого GUI-модуля, використовуваного із системами нечіткого логічного виведення, командою **Rules…** меню **Edit** або натисканням клавіш Ctrl+3. У FIS-редакторі відкрити редактор бази знань можна також подвійним щигликом лівою кнопкою миші по прямокутнику з назвою системи нечіткого логічного виведення, розташованого в центрі графічного вікна.

Загальний вид редактора бази знань із указівкою функціонального призначення основних областей графічного вікна наведений на рис. 5.

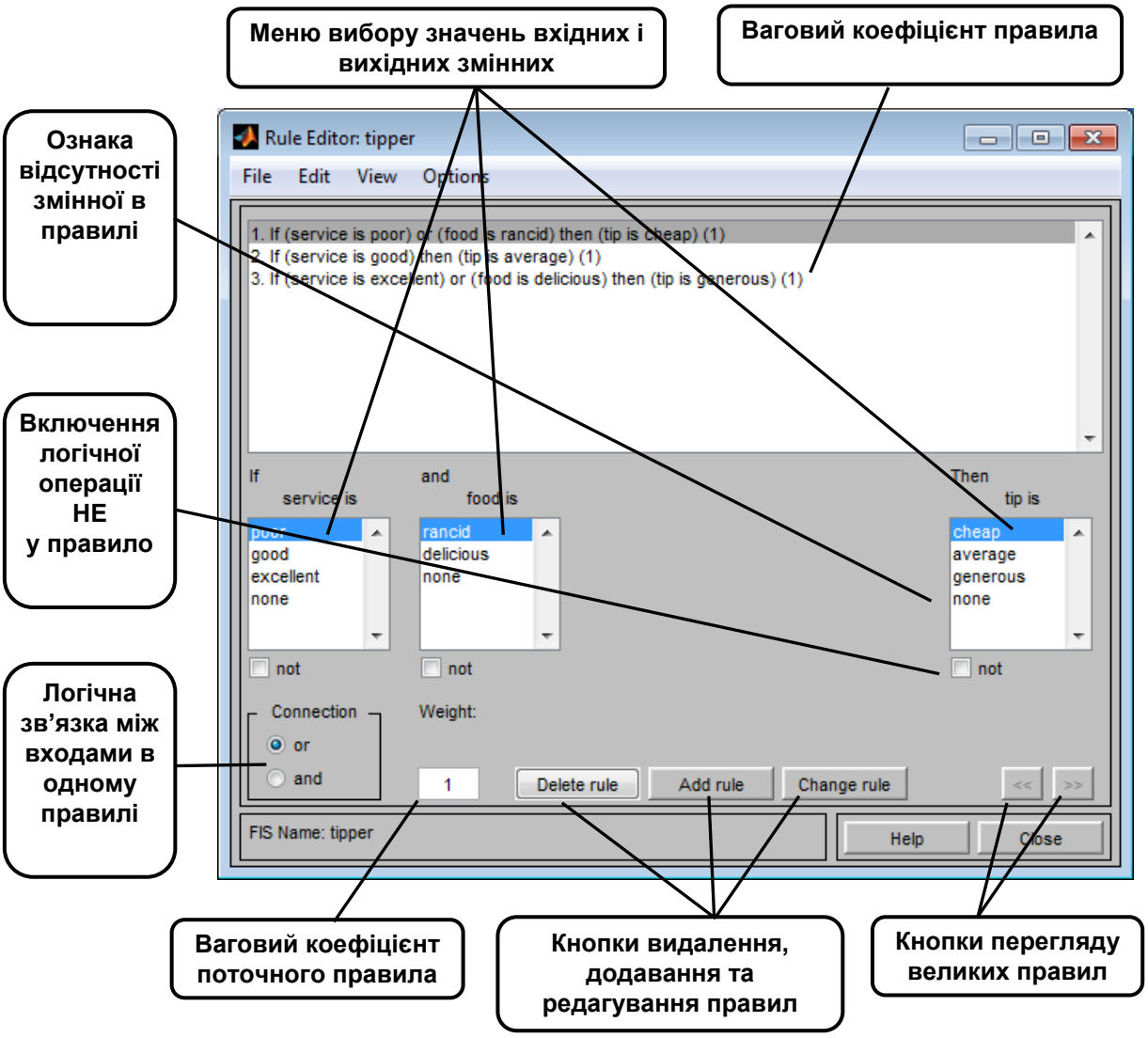


Рисунок 5 – Редактор бази правил (знань)

У нижній частині графічного вікна розташовані кнопки **Help** і **Close** для виклику вікна довідки і закриття редактора.

Редактор функцій належності містить чотири системних меню **File, Edit, View, Options,** меню вибору термів вхідних і вихідних змінних, поля установки логічних операцій **І, АБО, НЕ** і ваг правил, а також кнопки редагування і перегляду правил. Склад меню **File, Edit, View** наведений в попередніх розділах.

Меню Options

Language – встановлення мови текстових ідентифікаторів **English** (Англійська), **Deutsch** (Німецька), **Francais** (Французька).

Format – встановлення форматів правил бази знань: **Verbose** – лінгвістичний; **Symbolic** – логічний; **Indexed** – індексований.

Для введення нового правила в базу знань необхідно за допомогою миші вибрати відповідну комбінацію лінгвістичних термів вхідних і вихідних змінних, установити тип логічного зв'язування (**І** чи **АБО**) між змінними усередині правила, установити наявність чи відсутність логічної операції **НЕ** для кожної лінгвістичної змінний, увести значення вагового коефіцієнта правила і натиснути кнопку **Add Rule**. За замовчуванням установлені

наступні параметри: логічне зв'язування змінних усередині правила – **I**; логічна операція **HE** – відсутня; значення вагового коефіцієнта правила – 1.

Можливі випадки, коли істинність правила не змінюється при довільному значенні деякої вхідної змінної, тобто ця змінна не впливає на результат нечіткого логічного виведення в даній області факторного простору. Тоді значення цієї змінної, як лінгвістичне, необхідно встановити **none**.

Для видалення правила з бази знань необхідно зробити однократний щиглик лівою кнопкою миші на цьому правилі та натиснути кнопку **Delete Rule**. Для модифікації правила необхідно зробити однократний щиглик лівою кнопкою миші на цьому правилі, потім установити необхідні параметри правила і натиснути кнопку **Edit Rule**.

Візуалізація нечіткого логічного виведення

Візуалізація нечіткого логічного виведення здійснюється за допомогою GUI-модуля **Rule Viewer** (рис. 6).

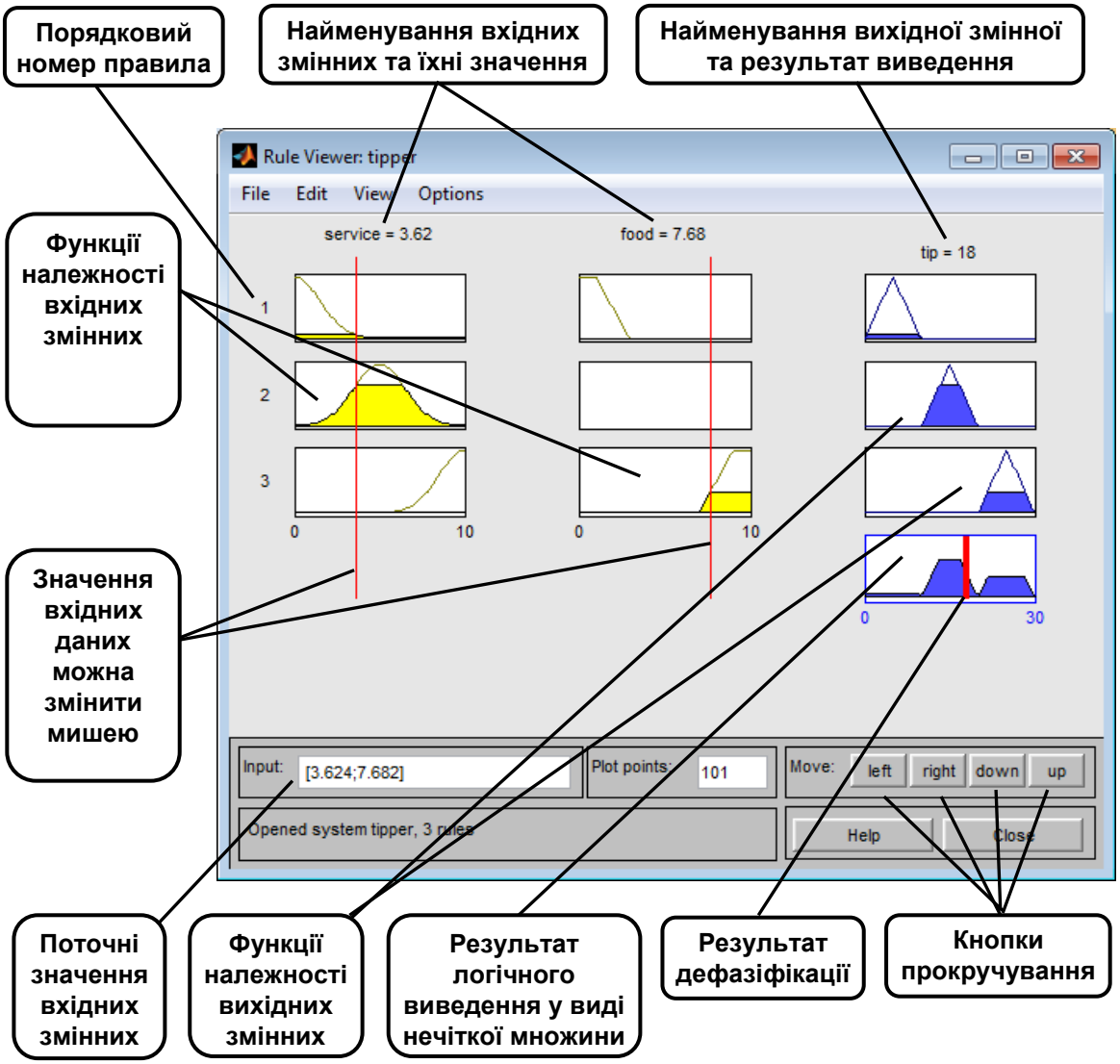


Рисунок 6 – Візуалізація логічного виведення за допомогою **Rule Viewer**

Цей модуль дозволяє проілюструвати хід логічного виведення за кожним правилом, одержання результуючої нечіткої множини і виконання процедури дефазіфікації. **Rule Viewer** може бути викликаний з будь-якого GUI-модуля, використовуваного із системами нечіткого логічного виведення, командою

View rules … (Ctrl+3) меню **View**.

Rule Viewer містить чотири меню – **File, Edit, View, Options,** дві області введення інформації – **Input і Plot points** та кнопки прокручування зображення вліво-вправо **left-right**, уверх-униз **up-down**. Склад меню **File, Edit, View** наведений в попередніх розділах. У нижній частині графічного вікна розташовані кнопки **Help** і **Close** для виклику вікна довідки і закриття редактора.

Кожне правило бази знань представляється у виді послідовності горизонтально розташованих прямокутників. При цьому перші два прямокутники відображають функції належності термів посилки правила (Якщо-частина правила), а останній третій прямокутник відповідає функції належності терму-наслідку вихідної змінної (То-частина правила).

Порожній прямокутник у візуалізації другого правила означає, що в цьому правилі посилання за змінною **food** відсутнє **food is none**. Жовте заливання графіків функцій належності вхідних змінних вказує наскільки значення входів, відповідають термам даного правила. Для виведення правила у форматі Rule Editor необхідно зробити однократний щиглик лівої кнопки миші по номері відповідного правила. У цьому випадку зазначене правило буде виведено в нижній частині графічного вікна.

Блакитне заливання графіка функції належності вихідної змінної являє собою результат логічного виведення у вигляді нечіткої множини за даним правилом. Результуючу нечітку множину, що відповідає логічному виведенню за всіма правилами, показано в нижньому прямокутнику останнього стовпця графічного вікна. У цьому ж прямокутнику червона вертикальна лінія відповідає чіткому значенню логічного виведення, отриманого в результаті дефазіфікації.

Уведення значень вхідних змінних може здійснюватися двома способами: шляхом уведення чисельних значень в область **Input** за допомогою миші, шляхом переміщення ліній-покажчиків червоного кольору.

В останньому випадку необхідно позиціонувати курсор миші на червоній вертикальній лінії, натиснути на ліву кнопку миші і не відпускаючи неї перемістити покажчик на потрібну позицію. Нове чисельне значення відповідної вхідної змінної буде перелічено автоматично і виведене у вікно **Input**. В області **Plot points** задається кількість точок дискретизації для побудови графіків функцій належності. Значення за замовчуванням – 101.

Перелік рекомендованої літератури

1. Штовба С.Д. Проектирование нечетких систем средствами MATLAB. – М: Горячая линия – Телеком, 2007, - 288с.: ил.
2. Пегат А. Нечеткое моделирование и управление; пер. с англ. / А. Пегат. – М.: БИНОМ, 2009. – 798 с.
3. Рутковская Д., Пилинский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Ж Пер. с польск. И.Д. Рудинского. М.: Горячая линия – Телеком, 2006. – 452с.: ил.

Практичне заняття 6

Тема: Моделювання обчислення логічних функцій на основі багатошарового персептрона

Мета. Отримати практичні навички створення і навчання багатошарового персептрона в пакеті MatLab.

Завдання

Створити й навчити нейронну мережу, здатну розв’язувати логічне завдання. Перевірити працездатність нейронної мережі.

Для розв'язку завдань, представлених у даному занятті, рекомендується використовувати нейронну мережу типу багатошаровий персептрон, що складається з трьох шарів: вхідний шар з 4-ма нейронами, прихований шар з двома нейронами і вихідний шар з одним нейроном (4-2-1). Порядок виконання роботи.

1. Скласти таблицю істинності для заданого логічного виразу (X_1, X_2, X_3, X_4, приймають значення 0 або 1).
2. Ввести в якості вхідних значень всі можливі сполучення X_1, X_2, X_3, X_4, а в якості вихідних значень – значення виходу в таблиці істинності.
3. Створити нейронну мережу з використанням операторів MATLAB.
4. Провести навчання нейронної мережі і вивести значення вагових коефіцієнтів.
5. Протестувати отриману нейронну мережу, задаючи в якості входу три варіанти значень X_1, X_2, X_3, X_4.
6. За результатами моделювання створити **.m** – файл з програмою.
7. Проаналізувати роботу нейронної мережі.

Таблиця 1 – Варіанти завдань

№ варіанту	Логічна функція
1	$(X_1 \vee X_2) \wedge X_3 \wedge X_4$
2	$X_1 \wedge X_2 \wedge (X_3 \vee X_4)$
3	$X_1 \wedge (X_2 \vee X_3) \vee X_4$
4	$X_1 \vee (X_2 \vee X_3) \wedge X_4$
5	$(X_1 \wedge X_2) \vee (X_3 \wedge X_4)$
6	$(X_1 \vee X_2) \wedge X_3 \vee X_4$
7	$X_1 \vee (X_2 \vee X_3) \wedge X_4$
8	$X_1 \wedge X_2 \vee X_3 \wedge X_4$
9	$X_1 \vee X_2 \wedge X_3 \vee X_4$

Для обчислення логічної функції можна скористатись таблицями 2 і 3.

Зміст звіту

1. Титульна сторінка.
2. Тема практичного заняття.
3. Мета заняття.
4. Номер варіанту та завдання до роботи.
5. Опис виконання завдань по пунктам з наданням рисунків і скріншотів.
6. Текст створеного *.m файлу.

7. Висновки, що відображують результати та їх критичний аналіз.

Контрольні питання

1. Модель штучного нейрона.
2. Основні типи активаційних функцій.
3. Спосіб обчислення коефіцієнтів w_{ij} в алгоритмах навчання.
4. Постановка задачі навчання ШНМ.
5. Параметри алгоритму навчання.
6. Підготовка даних для навчання ШНМ.

Таблиця 2 – Істинність операції І (∧) (∨)

X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Таблиця 3 – Істинність операції АБО

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

Теоретичні відомості

Штучна нейронна мережа (ШНМ) – набір елементарних нейроподібних перетворювачів інформації (нейронів), з'єднаних між собою.

На вхід формального нейрона надходить набір вхідних сигналів x_1, x_2, ..., x_n або вхідний вектор x. Кожен вхідний сигнал помножується на відповідну вагу w_1, w_2,..., w_n. Вага зв'язку є скалярною величиною, додатною для збудливих і від'ємною для гальмуючих зв'язків. Зважені вагами зв'язків вхідні сигнали надходять на блок сумації, де здійснюється їх алгебраїчне підсумовування та визначається рівень збудження нейроподібного елемента y:

$$y = f\left(\sum_{i=o}^{n} W_i x_i + \theta\right) \qquad (1)$$

де f – нелінійна функція активації, θ – деякий постійний зсув.

Для більшості моделей ШНМ такий нейрон є основним конструкційним елементом і часто представляється графічно рис. 1.

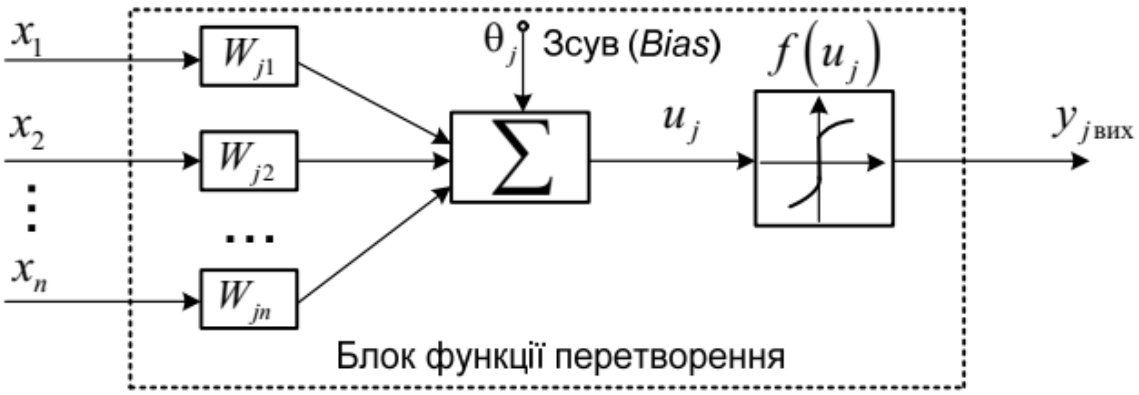


Рисунок 1 – Модель штучного нейрона (одношаровий персептрон)

Одношаровий персептрон є одним з найпростіших варіантів ШНМ і містить лише один нейрон.

В якості функції f за звичай використовуються найпростіші нелінійні функції:

бінарна (порогова): $$f(\mathrm{x}) = \begin{cases} 1, \text{ при } \mathrm{x} > 0, \\ 0, \text{ при } \mathrm{x} < 0, \end{cases} \quad (2)$$

або сигмоїдна: $$f(x) = \frac{1}{1+e^{-\alpha x}}. \quad (3)$$

Одним з типів нейронних мереж є мережі прямого поширення або багатошаровий персептрон рис. 2.

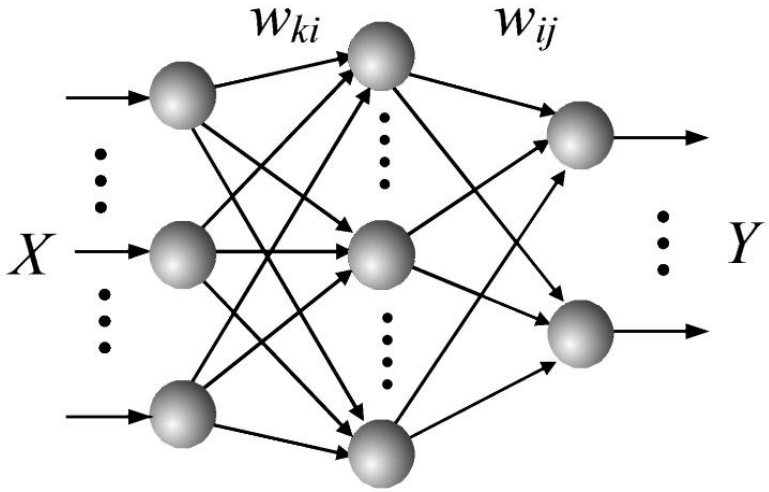


Рисунок 2 – Структура мережі прямого поширення
(багатошаровий персептрон)

Штучна нейронна мережа прямого поширення складається з вхідного шару, декількох схованих шарів і вихідного шару і описується формулою:

$$y_j = f\left(\sum_{i=o}^{n} W_{ji} x_i + \theta_j\right) \quad (4)$$

Навчання нейронної мережі.

ШНМ прямого поширення навчаються за методом навчання з учителем, коли для відомих даних на вході мережі ми знаємо, який результат повинен бути на виході. В такому випадку навчити ШНМ означає знайти такі вагові коефіцієнти w_{ji} і зсуви θ_j для кожного нейрону мережі з формули (4), щоб різниця між бажаними і обчисленими результатами на виході не перевищувала заданий наперед рівень помилки. В алгоритмах навчання ця помилка описується функцією втрат (loss function). Обчислення w_{ji} і θ_j здійснюється по крокам за формулами:

$$w_{ij}(n+1) = w_{ij}(n) - \beta \Delta w_{ij} \quad (5)$$

$$\theta_j(n+1) = \theta_j(n) - \beta \Delta \theta_j \quad (6)$$

На першому кроці ці значення обираються випадковим чином, тому повторне навчання на тих же даних буде давати різні результати. Спосіб обчислення значень Δw_{ji} і $\Delta\theta_j$ визначається алгоритмом навчання. Суттєве значення в цих формулах має коефіцієнт β, що зветься коефіцієнтом швидкості навчання. Його встановлюють у межах $0< \beta <1$ за звичай $\beta=0.1$. При великих значеннях β є ймовірність проскочити оптимальне значення ваг, при малих значеннях β навчання буде відбуватись довго. Досвід роботи з ШНМ показав, що не для любих даних мережу можна навчити, тому, щоб уникнути нескінченого циклу вводять обмеження на кількість епох. Епоха – це коли в алгоритмі навчання на вхід ШНМ поступають усі дані для навчання. Крім цього існує явище перенавчання, коли кажуть, що мережа не навчилась, а просто запам'ятала вхідні дані. Для виявлення цього явища навчальну вибірку розбивають на основну або тренувальну (*Train*), перевірочну (*Validation*) для виявлення перенавчання і тестову (*Test*) для оцінки якості навченої мережі. Перенавчання фіксується на кроці, коли функція втрат продовжує зменшуватись для тренувальної вибірки і починає збільшуватись для перевірочної. Крім цього для припинення навчання можна задавати максимальний час навчання і специфічні параметри алгоритмів навчання, наприклад, мінімальний рівень градієнту функції втрат, що використовується при обчисленні Δw_{ji} і $\Delta\theta_j$.

Створення нейронної мережі.

Вибір структури нейронної мережі являє собою окреме завдання й полягає у виборі топології мережі (кількості шарів і кількості нейронів в кожному шарі) й функцій активації кожного нейрона. За звичай функція активації однакова для всіх нейронів шару.

Створення навчальної вибірки. Навчальна вибірка повинна мати достатній розмір, щоб забезпечити ефективність навчання, враховуючи, що в алгоритмах навчання вона випадковим чином розбивається на тренувальну, перевірочну і тестову. Зазвичай співвідношення між Train, Validation та Test вибірками: 70: 20: 10.

Функція втрат (*loss function*) показує якість навчання нейронної мережі або ступінь відповідності множини виходів ШНМ бажаній множині виходів.

Епоха – крок навчання ШНМ, за який на вхід послідовно представляються усі вектори навчальної вибірки.

Приклад створення і навчання нейронної мережі

```
% Навчальна вибірка
x=[0 0 0 0 1 1 1 1;
   0 0 1 1 0 0 1 1;
   0 1 0 1 0 1 0 1];
y=[0 0 0 1 0 0 0 1];
% Структура і параметри нейронної мережі прямого поширення (newff)
% кількість шарів – 2;
% кількість нейронів у вхідному шарі – 3;
% кількість нейронів у схованому шарі – 2;
% функція активації нейронів вхідного і схованого шару – сигмоїда;
```

```
% мережа буде навчатися за алгоритмом Левенберга-Макварта
net=newff(x,y,[3,2],{'logsig','logsig'},'trainlm');
net.trainparam. show=25;        % кількість епох відображення на графіку
% значення функції втрат під час навчання
net.trainparam.lr= 0.1;         % коефіцієнт швидкості навчання
net.trainparam. epochs=500;     % максимальна кількість епох навчання
net.trainparam. goal=0.0001;    % значення похибки функції втрат
net.divideParam.trainRatio = 100/100; % параметри розподілу навчальної
net.divideParam.valRatio = 0/100;     % вибірки на тренувальну, валідаційну
net.divideParam.testRatio = 0/100;    % і тестову
net=train(net, x, y); % навчання мережі
w1=net.iw{1,1}; % вивід вагових коефіцієнтів і зсувів нейронів вхідного шару
display(w1);
b1=net.b{1,1};
display(b1);
w2=net.lw{2,1}; % вивід вагових коефіцієнтів і зсувів нейронів другого шару
display(w2);
b2=net.b{2,1};
display(b2);
a=sim(net, x); % обчислення помилки між бажаним значенням і обчисленим
display(a);
display(y);
er=a-y;
plot(er);
```

Після початку навчання з’являється вікно Neural Network Training (nntraintool) (рис. 4.), в якому відображається структура створеної нейронної мережі (панель Neural Network), деякі параметри алгоритму навчання (панель Algorithms), результуючі значення досягнутих параметрів навчання (панель Progress), результати навчання у виді графіків (панель Plots), Якщо навчання відбувається довго, то його можна призупинити, натиснувши кнопку Stop Training.

Процес зміни функції втрат під час навчання і результат можна спостерігати у вікні Neural Network Training Performance рис. 3, яке викликається з вікна Neural Network Training (nntraintool) натисканням кнопки Performance рис. 4.

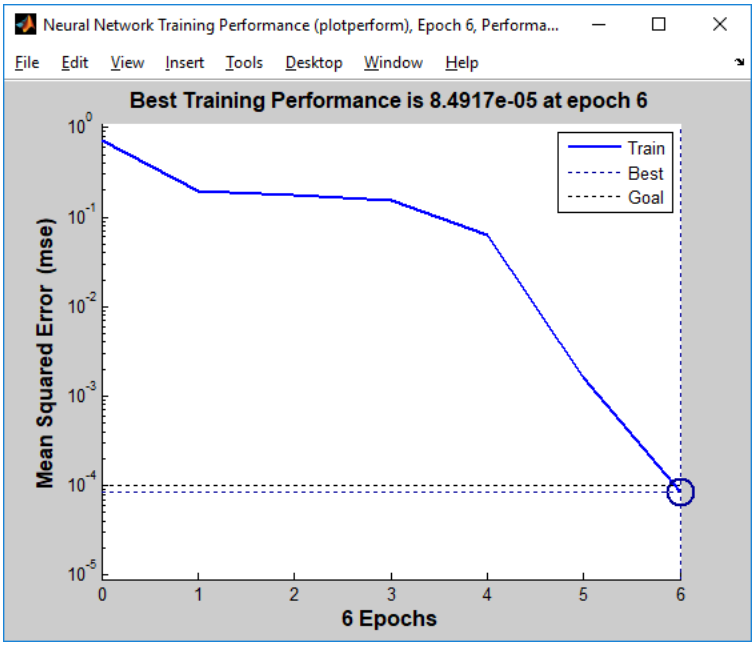


Рисунок 3 – Вікно зміни функції втрат при навчання нейронної мережі

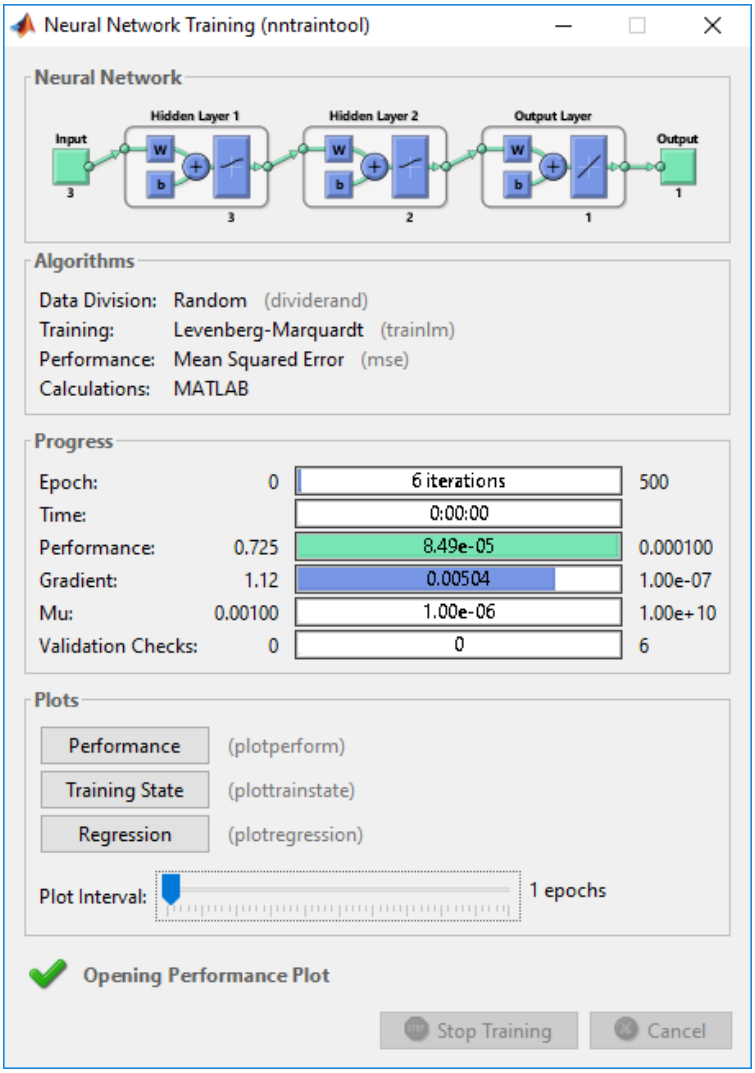


Рисунок 4 – Вікно контролю навчання нейронної мережі

На рис. 3 і рис. 4 представлений результат навчання мережі з наведеного вище прикладу. Функція newff примусово добавляє на виході один нейрон з лінійною функцією активації.

Таблиця 3 – Представлення структури нейромережі у MATLAB

Поле структури	Опис значення поля
numInputs	кількість входів мережі
numLayers	кількість (схованих) шарів мережі без урахування вхідного шару
biasConnect, inputConnect, layerConnect, outputConnect, targetConnect	булеві масиви, що визначають зв'язки між структурними елементами мережі
numOutputs	кількість виходів мережі
numTargets	кількість цільових ознак
numInputDelays	кількість затримок вхідного шару
numLayerDelays	кількість затримок схованих шарів
inputs	входи мережі
layers	шари мережі
outputs	виходи мережі
targets	цільові ознаки
biases	масив порогів нейронів мережі
inputWeights	масив вагових коефіцієнтів вхідного шару мережі
layerWeights	масив вагових коефіцієнтів схованих шарів мережі
adaptFcn	ім'я функції адаптації нейронів
initFcn	ім'я функції ініціалізації мережі
performFcn	ім'я цільової функції навчання (помилки)
trainFcn	ім'я функції, що реалізує процес навчання
adaptParam	параметри адаптації мережі
initParam	параметри ініціалізації мережі
performParam	параметри цільової функції мережі
trainParam.epochs	максимально допустима кількість ітерацій навчання (епох)
trainParam.goal	максимально допустиме значення цільової функції навчання
trainParam.max_fail	максимально допустима кількість відмов у процесі навчання мережі
trainParam.mem_reduc	коефіцієнт, що регулює (зменшує) використання пам'яті при навчанні нейромереж
trainParam.min_grad	мінімально допустиме значення градієнту цільової функції
trainParam.mu, trainParam.mu_dec, trainParam.mu_inc, trainParam.mu_max	параметри методу Левенберга-Марквардта
trainParam.show	кількість ітерацій, через яку будуть відображатися зміни стану процесу навчання мережі
trainParam.time	максимально допустимий час навчання мережі у секундах
iw	масив значень ваг вхідного (першого) шару
lw	масив значень ваг схованих шарів
b	масив значень порогів нейронів
userdata	дані користувача
divideParam.trainRatio	розмір навчальної вибірки відносно всіх початкових даних (приклад: 70/100)
divideParam.valRatio	розмір перевірочної вибірки відносно всіх початкових даних (приклад: 20/100)
divideParam.testRatio	розмір тестової вибірки відносно всіх початкових даних (приклад: 10/100)

Перелік рекомендованої літератури

1. Ямпольський Л.С. Нейротехнології та нейрокомп’ютерні системи: підручник / Л.С. Ямпольський, О.І. Лісовиченко, В.В. Олійник. – К.: «Дорадо-Друк», 2016. – 576 с.: іл.

2. Дьяконов В. П. MATLAB. Полный самоучитель. – М.: ДМК Пресс, 2012. – 768 с.: ил.

3. Хайкин, Саймон Нейронные сети: полный курс, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 1104 с.: ил. – Папал. Тит. Англ.

4. Бодянский Е.В., Руденко О.Г. Искусственные нейронные сети, обучение, применения. Монография. Харьков: ТЕЛЕТЕХ, 2004. – 372 с.: ил.

Практичне заняття 7

Тема: Створення мережі прямого поширення засобами nntool

Мета. Придбання навичок застосування командно-графічного інтерфейсу системи для побудови та дослідження нейронних мереж різної архітектури Отримати практичні навички створення і навчання мережі прямого поширення засобами nntool.

Завдання

1. Для даних з практичного заняття 6 створити нейронну мережу типу Feed-forward backprop засобами nntool.
2. Провести навчання нейронної мережі і вивести значення вагових коефіцієнтів.
3. Протестувати отриману нейронну мережу, задаючи в якості входу три варіанти значень X_1, X_2, X_3, X_4.
4. Зберегти створену мережу у файлі формату **.mat**.
5. Проаналізувати роботу нейронної мережі і порівняти результати з практичного заняття 5.

Зміст звіту

1. Титульна сторінка.
2. Тема практичного заняття.
3. Мета заняття.
4. Завдання.
5. Опис виконання завдань по пунктам з наданням рисунків і скріншотів.
6. Висновки.

Контрольні питання

1. Модель штучного нейрона.
2. Основні типи активаційних функцій.
3. Спосіб обчислення коефіцієнтів w_{ij} в алгоритмах навчання.
4. Постановка задачі навчання ШНМ.
5. Параметри алгоритму навчання.
6. Підготовка даних для навчання ШНМ.

Теоретичні відомості

MATLAB містить візуальний інтерфейсний модуль nntool, який входить до бібліотеки Neural Network Toolbox.

Використання nntool дозволяє більш зручними засобами, ніж написання програми вручну, будувати нейромережеві моделі об'єктів та процесів. Розглянемо деякі основні можливості та прийоми роботи із засобом nntool.

Після запуску MATLAB в командному вікні для початку роботи із nntool треба ввести: nntool. Після цього завантажиться робоча панель nntool **Network/Data Manager** (рис. 1).

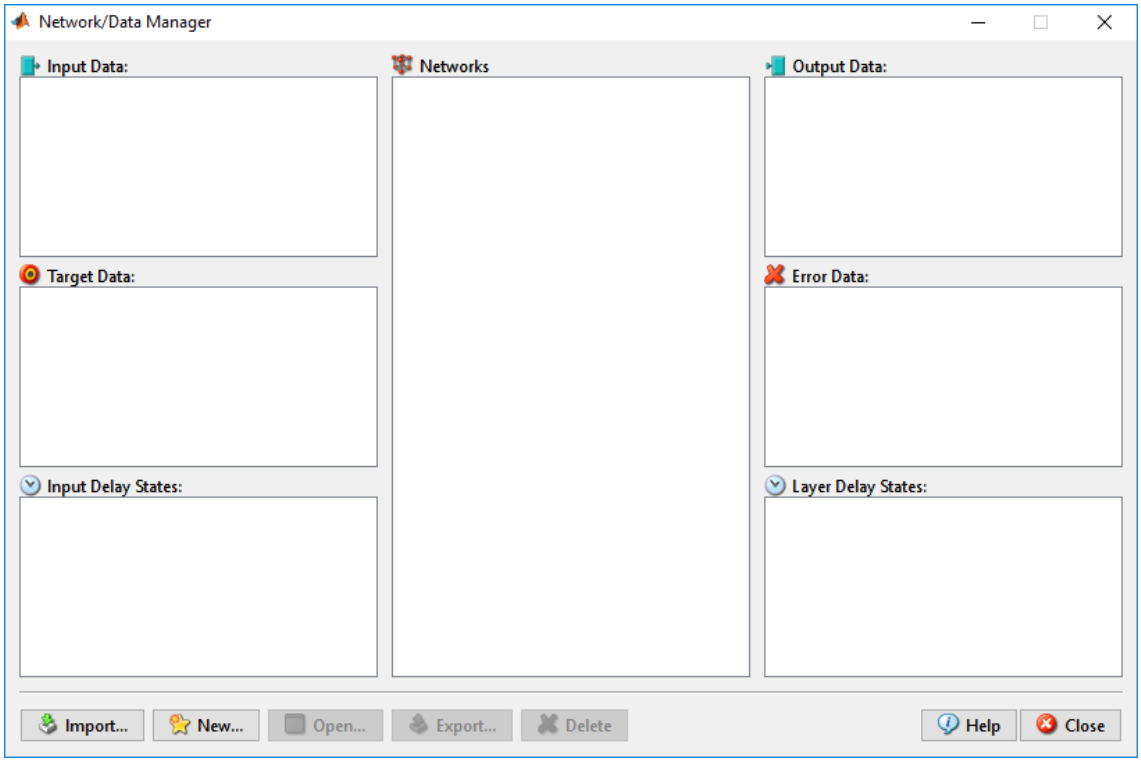


Рисунок 1 – Головна діалогова форма засобу nntool

На панелі **Network/Data Manager** користувач має натиснути кнопки для завдання початкових даних для побудови нейромережевої моделі:

Import – дані імпортуються з робочого простору MATLAB або файлу;

New…– створення нових даних і мережі;

Open – відкриття створеної мережі з ім'ям на панелі **Networks** (рис. 1);

Export – збереження даних мережі в робочому просторі MATLAB або файлі;

Delete –видалення непотрібного елементу даних (змінної або мережі).

Для створення нової мережі користувач має натиснути кнопку **New…,** яка викликає редактор **Create Network or Data** (рис. 2).

Створення нейронної мережі потрібно починати з створення вхідних і вихідних даних. Для цього треба вибрати закладку **Data** (рис. 2). Вхідні і вихідні дані для MATLAB є змінними, тому треба дати їм ім'я (панель **Name**), визначити чисельні значення (панель **Value**) і тип даних (панель **Data Type**). Для вхідних даних треба обрати тип **Inputs**, натиснути кнопку **Create** і підтвердити операцію. Для вихідних даних треба обрати тип **Targets**,

натиснути кнопку **Create** і підтвердити операцію

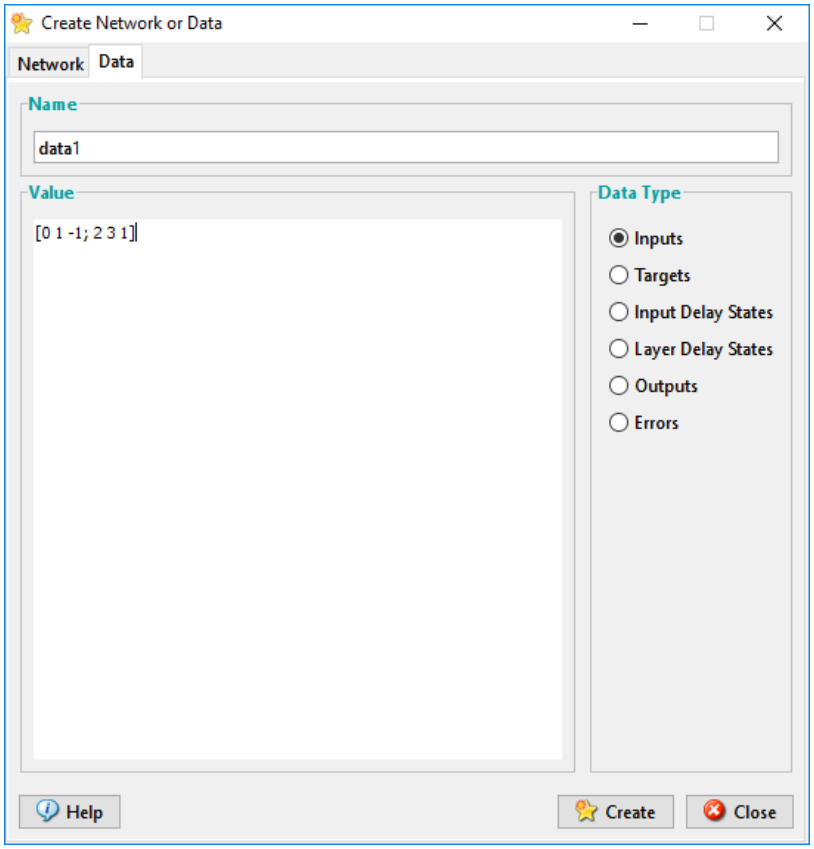


Рисунок 2 – Редактор даних засобу nntool

Якщо дані уже існують у вигляді зовнішніх файлів або містяться у середовищі MATLAB у вигляді змінних, вони можуть бути імпортовані за допомогою кнопки **Import** (рис. 1). При цьому з'являється діалогова форма **Import to Network/Data manager** (рис. 3).

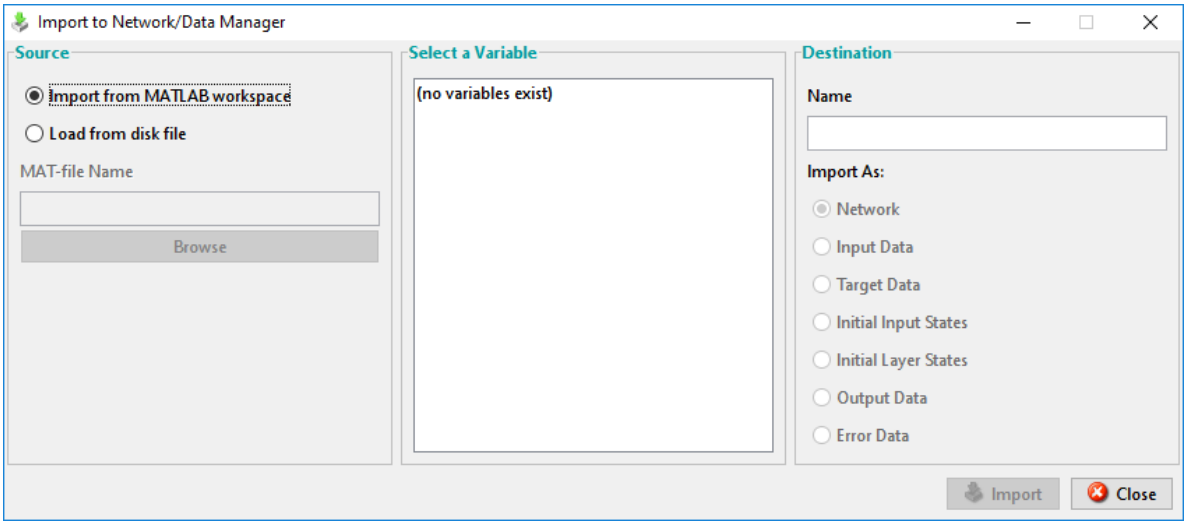


Рисунок 3 – Діалогова форма імпорту даних

Панель **Source** дозволяє обрати джерело введення даних:

Import from MATLAB workspace (імпорт даних із середовища MATLAB);

Load from disk file (завантаження даних із файлу на диску).

Кнопка **Browse** дозволяє обрати необхідний файл. Панель **Select a**

Variable дозволяє вказати засобу nntool, яку змінну треба використовувати для імпорту даних.

Панель **Destination** дозволяє задати змінну для прийому даних, що імпортуються. Її ім'я вказується в панелі **Name**, а призначення (**Import as)** обирається із наведеного меню.

Кнопка **Export** головної діалогової форми (рис. 1) дозволяє зберегти дані із середовища nntool у файлі на диску, або передати їх до середовища MATLAB.

Для створення мережі та визначення її параметрів необхідно вибрати закладку **Network** (рис. 4).

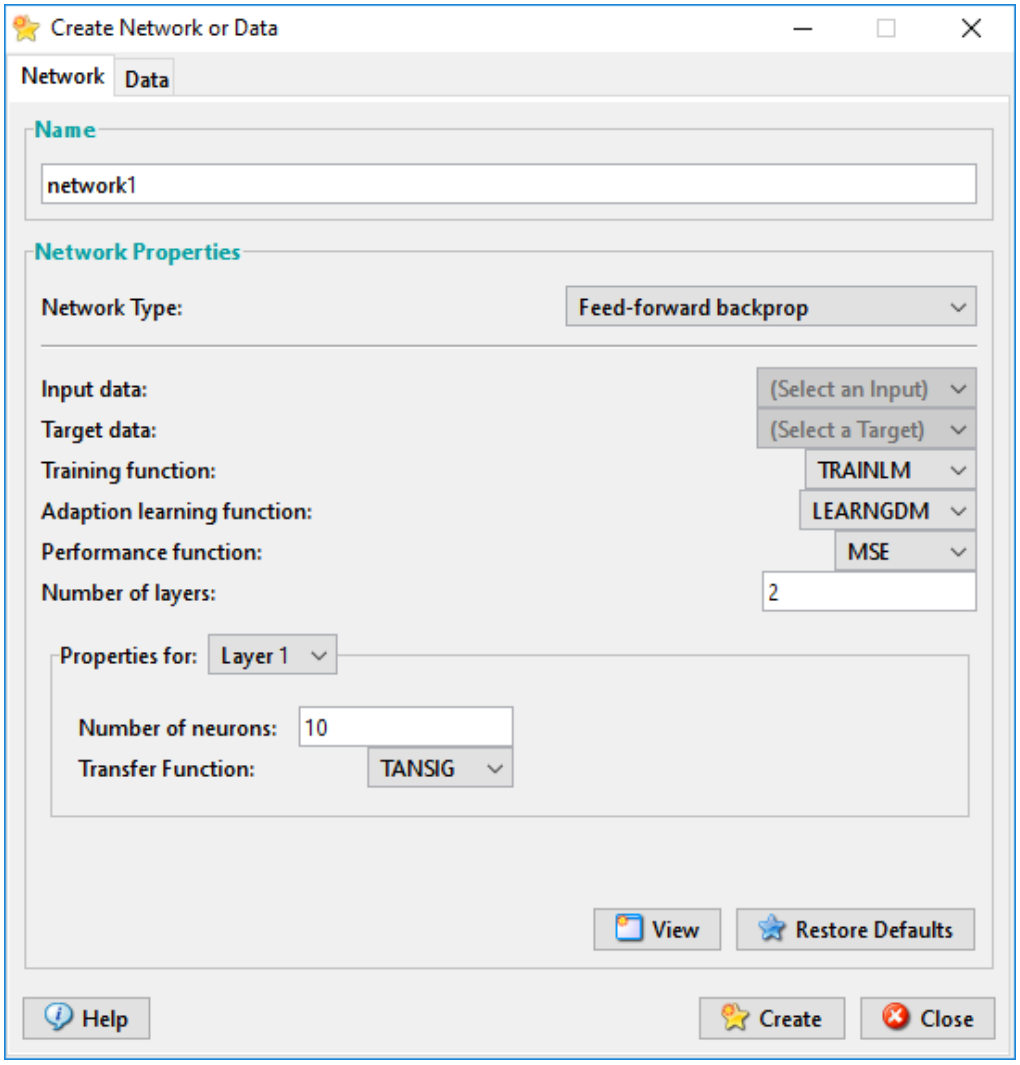


Рисунок 4 – Форма конструювання нейромережі

В панелі **Name** визначається ім'я змінної для збереження мережі. На панелі **Network Properties** визначаються параметри мережі:

Network Type дозволяє обрати тип архітектури мережі (наприклад, **Feed-forward backprop** – мережа прямого поширення);

Input data – вибір імені вхідних даних мережі, створених раніше;

Target data – вибір імені вихідних даних мережі, створених раніше;

Training Function – вибір алгоритму навчання мережі;

Adaptation Learning Function – вибір алгоритму адаптації навчання мережі;

Performance Function – вибір методу оцінки функції втрат (**MSE** – метод найменших квадратів);

Number of Layers – визначення кількості шарів мережі.

Панель **Properties for Layer K** дозволяє задати властивості для нейронів К-го шару мережі:

Number of Neurons – кількість нейронів для поточного шару;

Transfer Function – тип функції активації нейронів поточного шару.

Кнопка **View** дозволяє отримати графічне зображення структури створеної нейромережі (рис. 5). Кнопка **Restore Defaults** відновлює параметри мережі за замовчуванням. Кнопка **Help** дозволяє викликати довідкову службу MATLAB з описом необхідних компонентів та поясненнями щодо їхнього використання.

Після створення мережі необхідно натиснути кнопку **Create.**

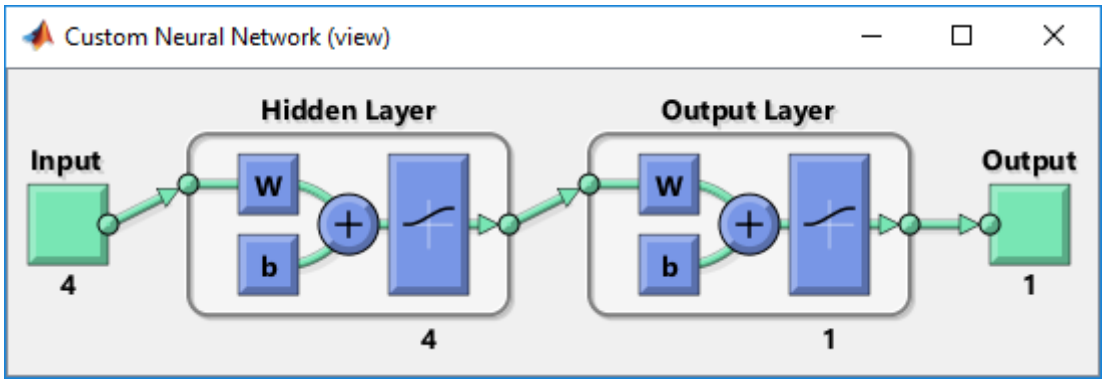


Рисунок 5 – Структура нейромережі

Після побудови нейромережі у нижній частині головної діалогової форми nntool (рис. 1) стає доступною панель **Networks only**, що призначена для роботи із побудованою мережею (рис. 6).

При натисненні будь-якої з кнопок цієї панелі викликається діалогова форма **Network**, яка містить набір закладок-панелей для роботи з мережею (рис. 6-10).

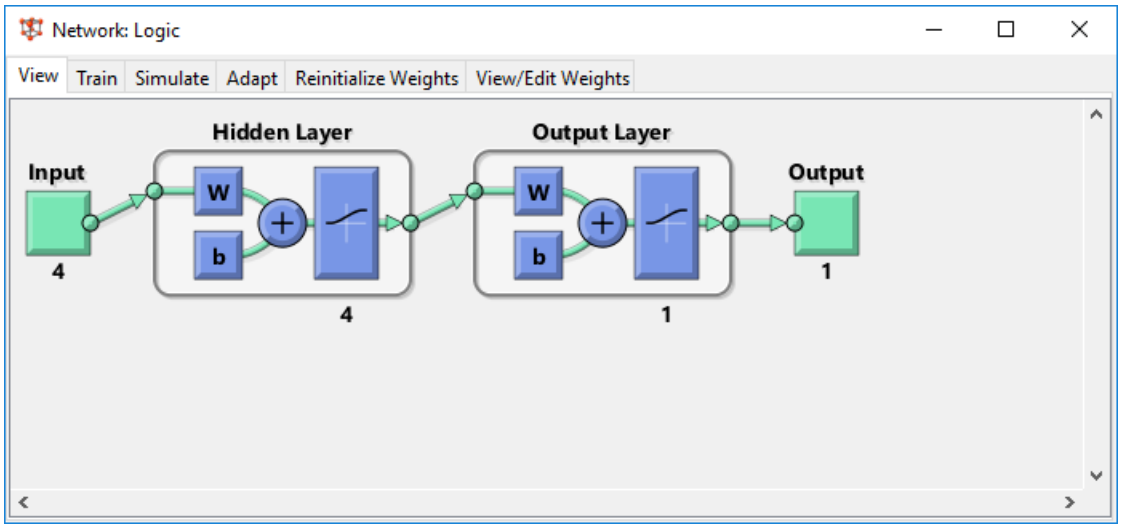


Рисунок 6 – Панель нейромережі. Закладка **View** – структура мережі.

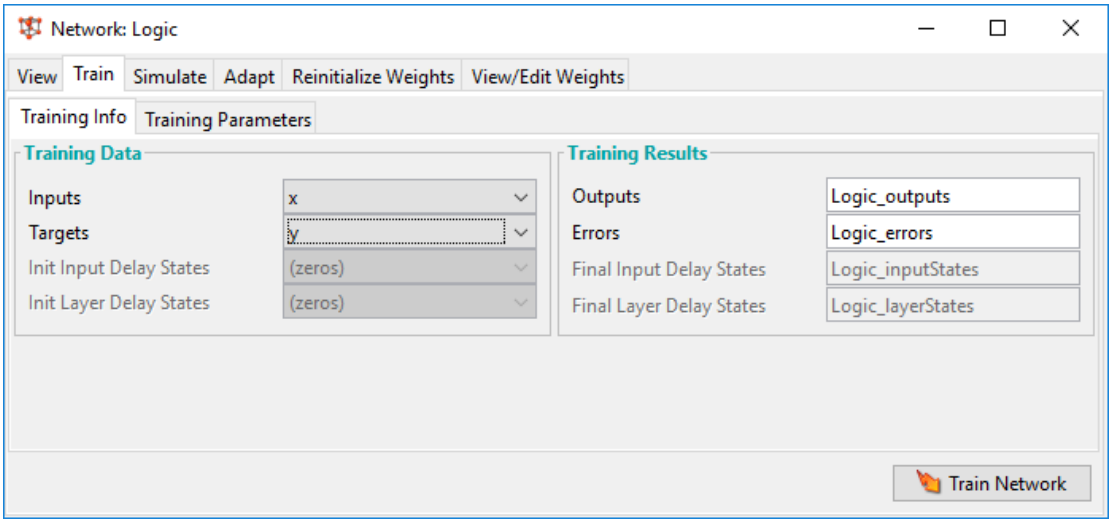


Рисунок 7 – Панель нейромережі (закладка **Train** – навчання мережі).

Закладка **Train** – має закладки **Training Info** (рис.7), де в панелі **Training Data** треба обрати імена вхідних (**Inputs)** і вихідних (**Outputs**) вихідних даних мережі.

На закладці **Training Parameters** (рис.8) можна змінити критерії припинення навчання і параметри алгоритму навчання (наприклад, алгоритму Левенберга-Макварта (рис. 8)).

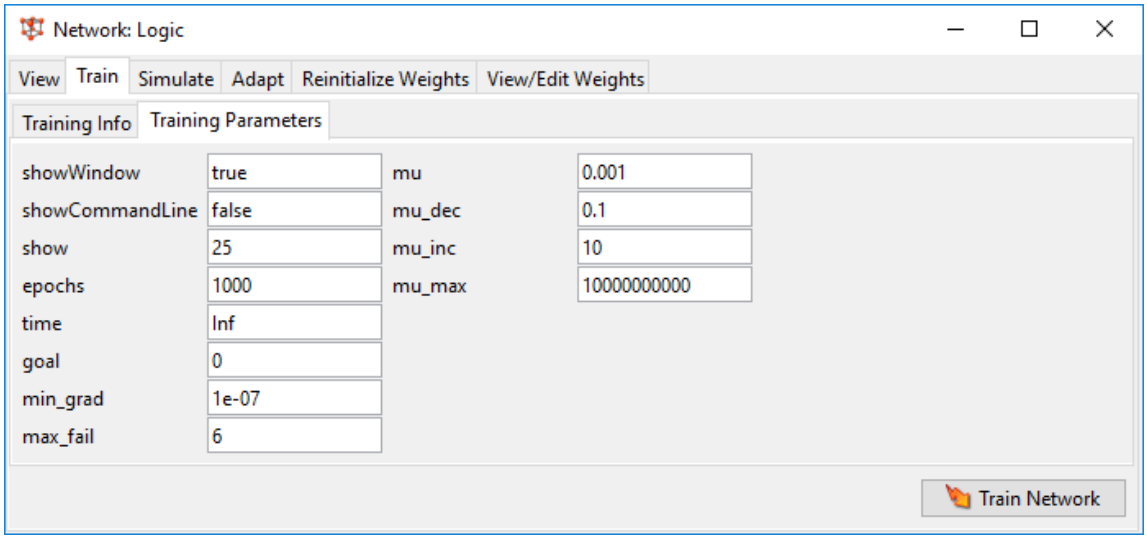


Рисунок 8 – Панель нейромережі (закладка **Training Parameters**).

Серед параметрів навчання, доступних на цій закладці обов’язково необхідно задати: **goal** – максимально припустиме значення функції втрат, **epochs** – максимальна припустима кількість циклів навчання мережі, **show** – шаг виводу на екран інформації про навчання мережі, задається в циклах навчання.

Закладка **Simulate** призначена для практичної роботи з навченою мережею. На закладці **View/Edit Weights** (рис. 9) можна вивести чисельні значення вагових коефіцієнтів і зсувів нейронів навченої мережі.

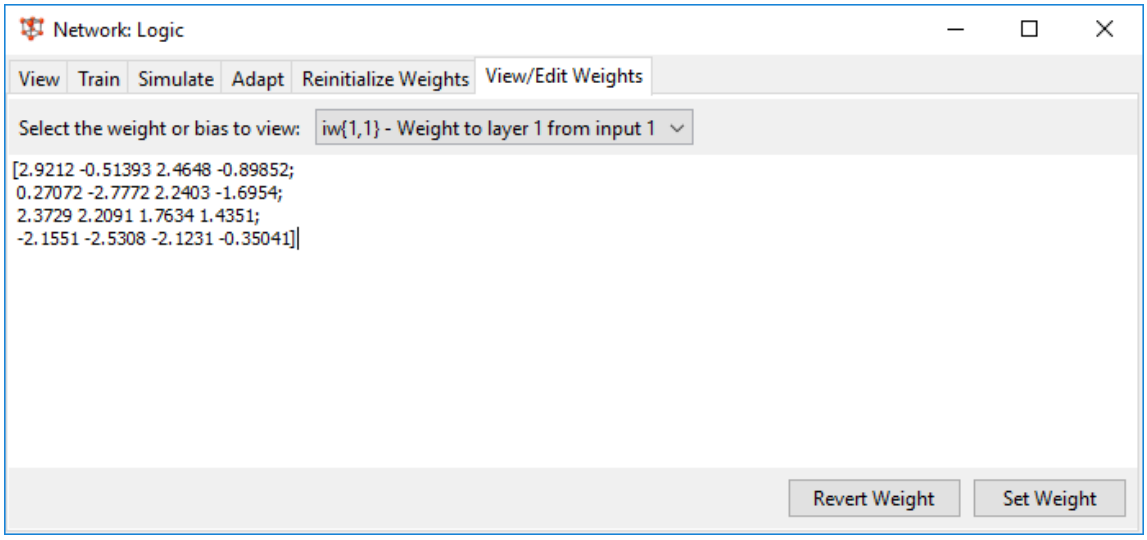


Рисунок 9 – Панель нейромережі. Закладка **View/Edit Weights**.

Структура і результати навчання представляються на панелі **Neural Network Training (nntraintool)** (рис. 10).

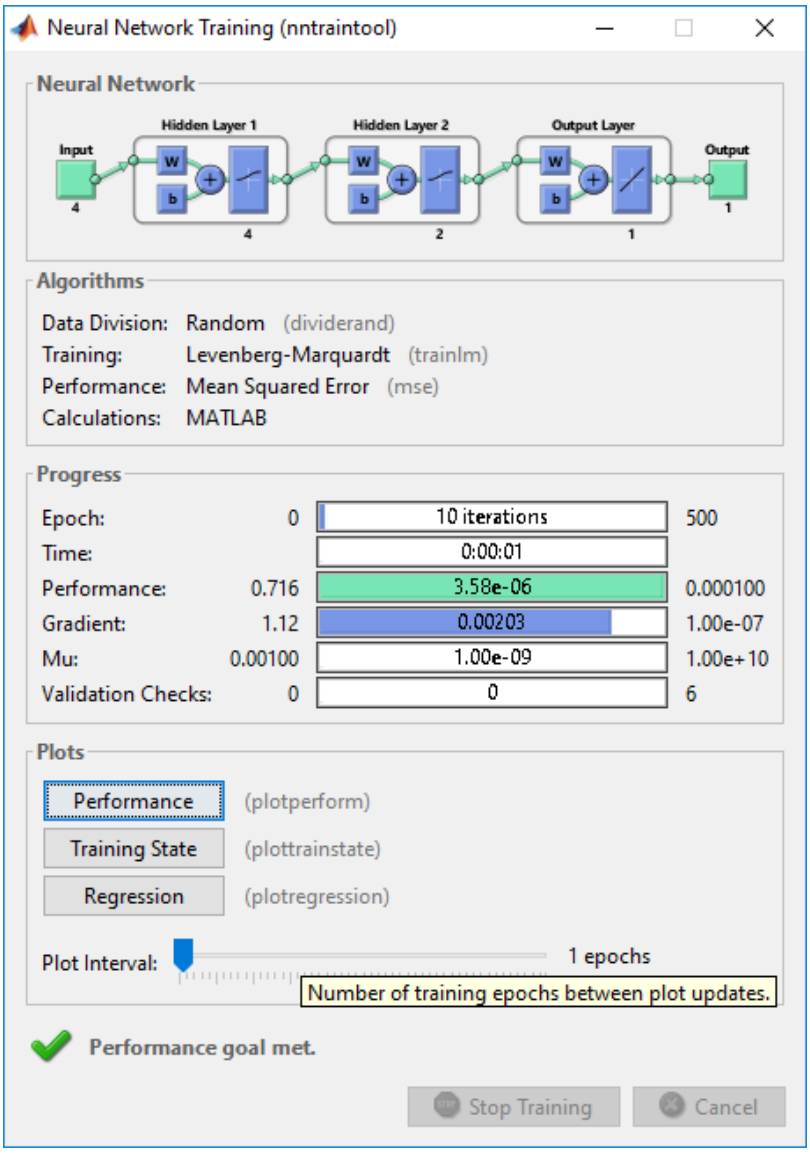


Рисунок 10 – Вікно контролю навчання нейронної мережі

В процесі навчання середовище MATLAB будує графік зміни значення

функції втрат по епохах (рис. 11). Для відкриття форми треба натиснути кнопку **Performance** (рис. 10). При натисненні кнопки **Training State** виводяться параметри алгоритму навчання по епохах.

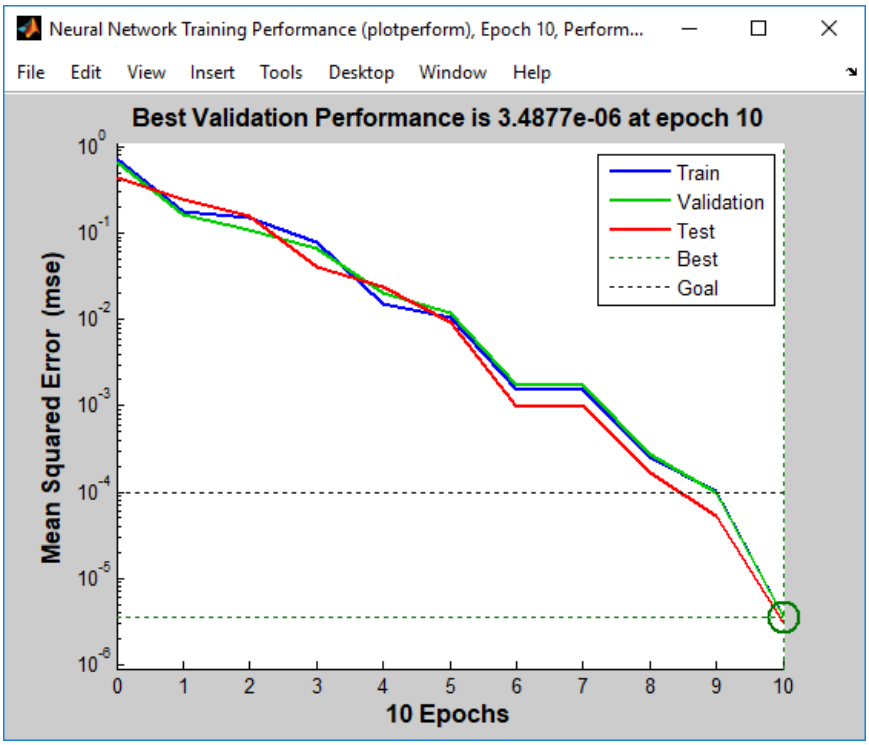


Рисунок 11 – Вікно контролю навчання нейронної мережі

Перелік рекомендованої літератури

1. Ямпольський Л.С. Нейротехнології та нейрокомп'ютерні системи: підручник / Л.С. Ямпольський, О.І. Лісовиченко, В.В. Олійник. – К.: «Дорадо-Друк», 2016. – 576 с.: іл.

2. Дьяконов В. П. MATLAB. Полный самоучитель. – М.: ДМК Пресс, 2012. – 768 с.: ил.

3. Хайкин, Саймон Нейронные сети: полный курс, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 1104 с.: ил. – Папал. Тит. Англ.

4. Бодянский Е.В., Руденко О.Г. Искусственные нейронные сети, обучение, применения. Монография. Харьков: ТЕЛЕТЕХ, 2004. – 372 с.: ил.

Практичне заняття 8

Тема: Застосування нейронних мереж для апроксимації функцій

Мета. Придбати практичні навички дослідження можливостей застосування нейронних мереж для вирішення прикладних завдань і взаємодію між застосунками MATLAB і робочим середовищем MATLAB.

Завдання

1. Побудувати модель нейронної мережі типу Feed-forward-backprop, що апроксимує поліном виду

$$Y = ax_1^2 + bx_2^2 + cx_1x_2 + dx_1 + ex_2 + f$$

на інтервалі [-1, 1], коефіцієнти полінома задані в табл.1. Варіант відповідає порядковому номеру за журналом.

2. Обґрунтувати вибір структури і параметрів мережі.
3. Навчити мережу і побудувати графік помилки мережі.
4. Зберегти навчену мережу у файлі.
5. Побудувати модель нейронної мережі типу Feed-forward-backprop для апроксимації нелінійної функції *f(t)* на проміжку [0, 10], заданої згідно з номером варіанту (табл.2).
6. Зберегти навчену мережу у файлі.
7. Побудувати на одному графіку криві бажаних і обчислених значь на виході навченої мережі.

Примітка. Іменем мережі повинно бути прізвище студента латинськими символами.

Таблиця 1 – Коефіцієнти полінома двох змінних

№	*a*	*b*	*c*	*d*	*e*	*f*	№	*a*	*b*	*c*	*d*	*e*	*f*
1.	-2	1	-1	2	1	-2	11.	-1	1	2	0	0	0
2.	-1	3	-1	2	-1	-1	12.	1	1	1	1	1	1
3.	-2	-2	2	2	-2	2	13.	2	1	1	2	2	-8
4.	-3	2	-2	3	-1	1	14.	0	1	0	1	0	-2
5.	-2	2	-2	2	-2	2	15.	1	0	1	0	1	-3
6.	-1	-1	-1	1	1	1	16.	2	1	-2	1	0	-7
7.	0	1	0	2	0	-3	17.	1	2	3	-3	-4	-1
8.	2	0	2	3	1	0	18.	2	1	2	-1	-3	-2
9.	2	-2	4	-4	3	-3	19.	2	1	1	-2	-2	-1
10.	-5	-3	3	3	2	1	20.	-1	-1	-1	-1	2	2

Таблиця 2 – Нелінійні функції для апроксимації

№	$f(t)$	№	$f(t)$
1.	$\cos^2(t)\sin(2t)$	2.	$t^2\sin(t)$
3.	$\sin(0.5t^2)$	4.	$\sin(t^2-10t)$
5.	$t\sin(t)$	6.	$\sin(t^2-5t)$
7.	$\cos(1.4^t)$	8.	$\sin(t^2-8t)$
9.	$\ln(t)\sin(t)$	10.	$\sin(t^2-12t)$
11.	$\ln(2t)\sin(t^{1.5})$	12.	$\sin(t^2-8t)$
13.	$0.01t^2\sin(t)$	14.	$\sin(t^2-12t)$
15.	$t\sin(2t)$	16.	$\sin(t^2-6t)$
17.	$t\sin(3t)$	18.	$\sin(t^2-4t)$
19.	$t\sin(4t)$	20.	$(t^2-10t)\sin(t^2-10t)$

Зміст звіту

1. Титульна сторінка.
2. Тема практичного заняття.
3. Мета заняття.
4. Завдання.
5. Опис виконання завдань по пунктам з наданням рисунків і скріншотів.
6. Висновки.

1. Приклад апроксимації функції двох змінних.

Нехай необхідно апроксимувати на інтервалі [0, 1] з кроком 0,1 поліном виду:

$$Y = 2x_1^2 - x_2^2 + 3x_1x_2 - 1$$

(a=2, b= –1, c=3, d=0, e=0, f=1)

Для формування вектора навчальних прикладів у вікні команд MATLAB необхідно набрати наступні команди:

```
x = 1;
for i = 0: .1: 1;
   for j = 0: .1: 1;
      x1x2 (1, x) = i;
      x1x2 (2, x) = j;
      T (x) = 2*i*i -j*j + 3*i*j -1;
      x = x + 1;
   end;
end;
```

Після чого будуть сформовані матриця вхідних даних *x1x2* і вектор цілей *T*. Імпортуємо дані з робочої області: вектор цілей *T* і матрицю вхідних даних *x1x2* (рисунок 1). Побудуємо за допомогою **Network/Data Manager** інструментального засобу Neural Network Tools модель нейронної мережі.

На першому кроці в **Network/Data Manager** виберемо опцію **Import** і визначимо вхідні і вихідні дані мережі за допомогою імпорту даних з робочого середовища MATLAB (рис. 1).

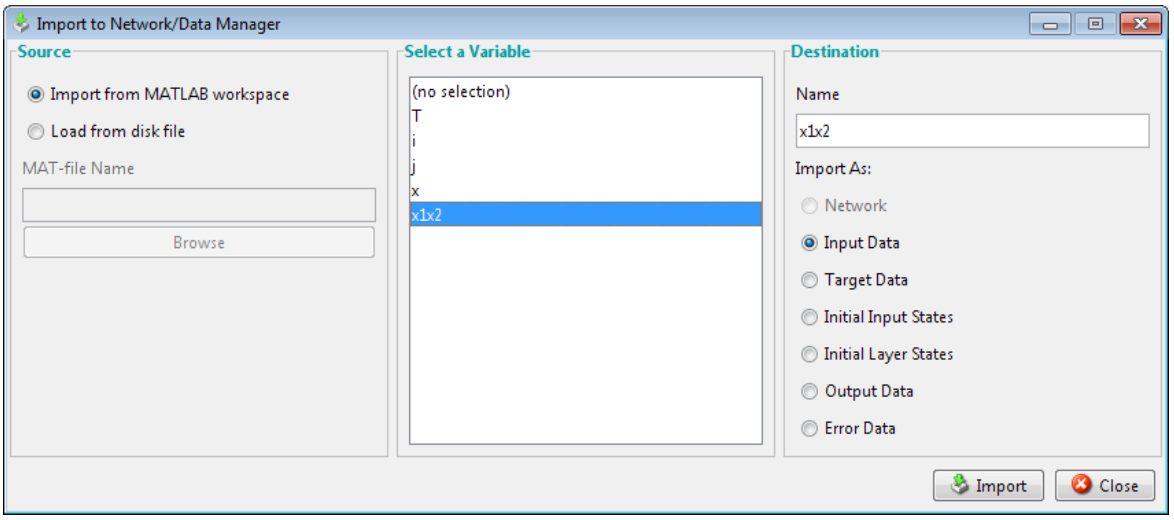


Рисунок 1. – Вікно імпорту і завантаження даних

Повернемось в **Network/Data Manager** і визначимо параметри нейронної мережі, обравши опцію **New…** і закладку **Network** (рис. 2).

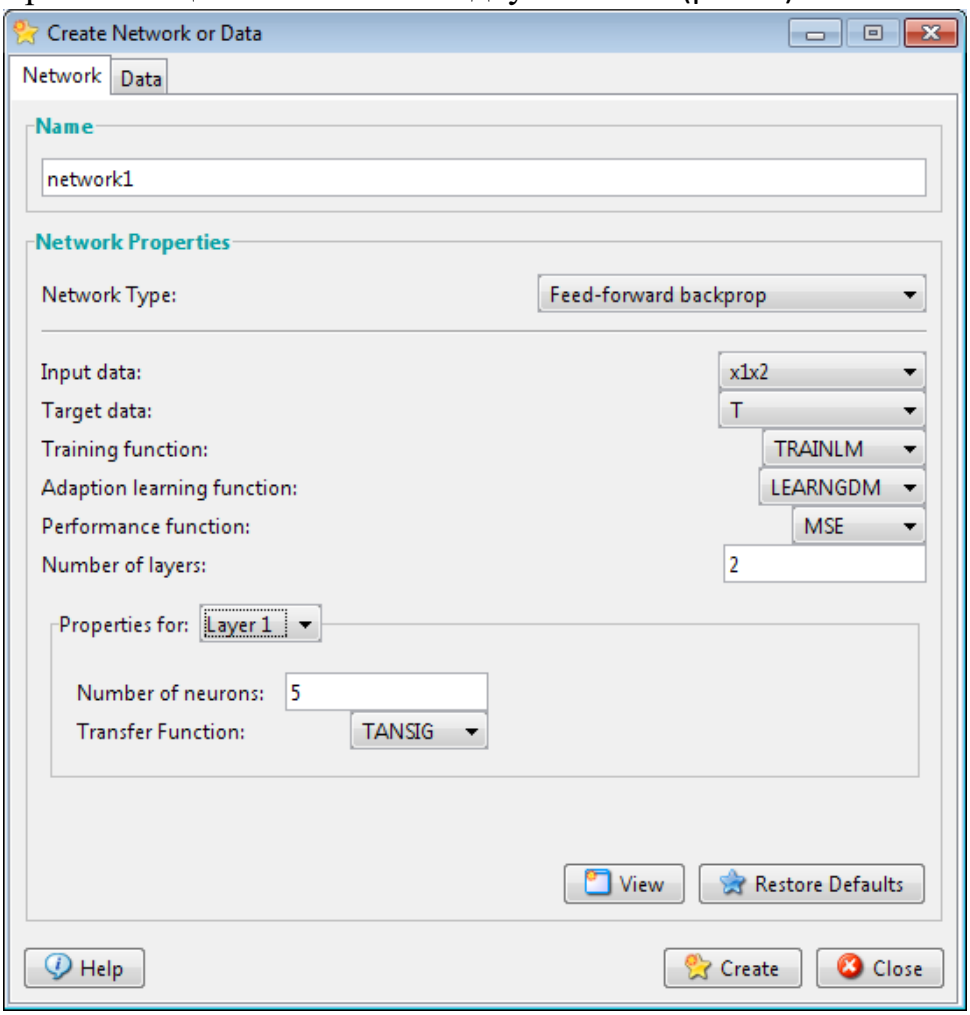


Рисунок 2 – Вікно створення нейронної мережі з ім’ям **network1**

Встановлені такі параметри: тип обраної мережі – Feed-forward-backprop; вхідні дані – матриця *x1x2;* вихідні дані – вектор *T;* кількість нейронів в першому шарі – 5, у другому 1; функція активації нейронів першого шару – будь-яка нелінійна (обрана TANSIG), другого – лінійна (PURELIN); алгоритм навчання – Левенберга-Маркардта (TRAINLM) (рис. 2). Ім'я мережі – **network1**. Після визначення параметрів мережі обираєм опцію **Create** і підтверджуємо її створення.

Перейдемо в **Network/Data Manager** (рис. 3)і відкриємо мережу оцією Open (рис. 4).

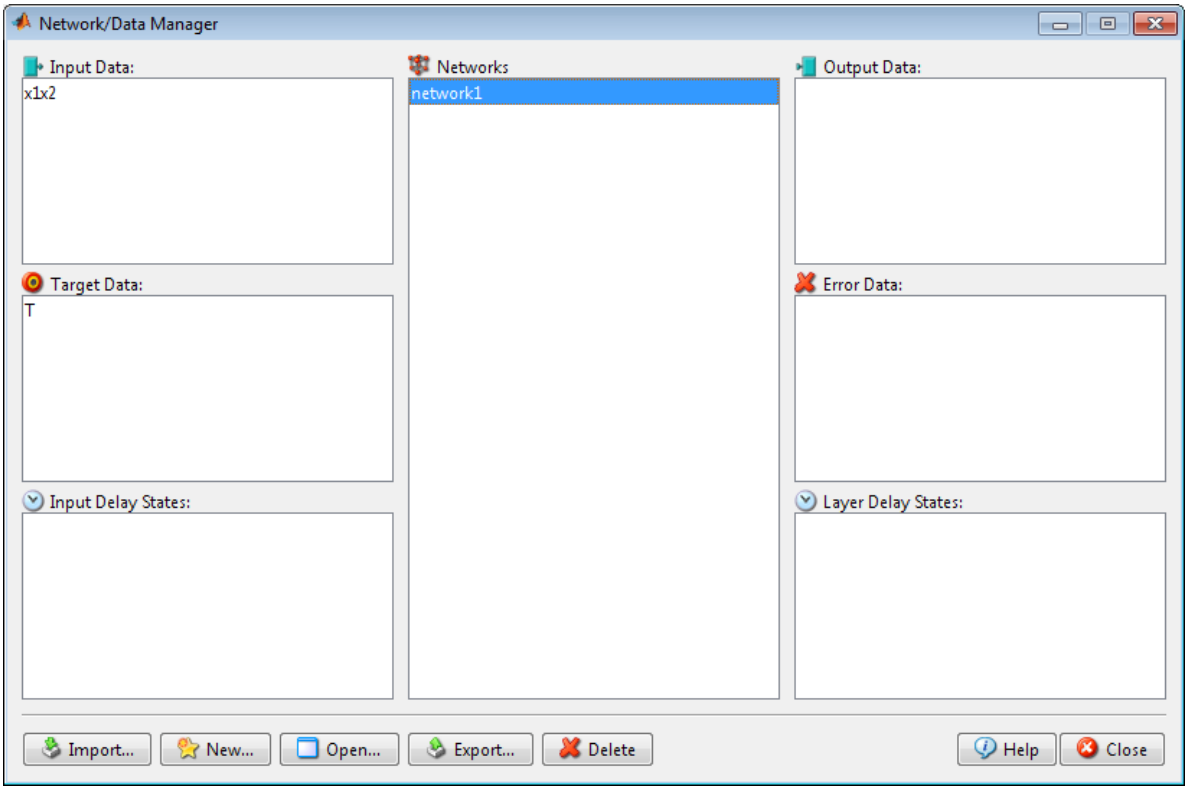


Рисунок 3 – Діалогова панель **Network/Data Manager**

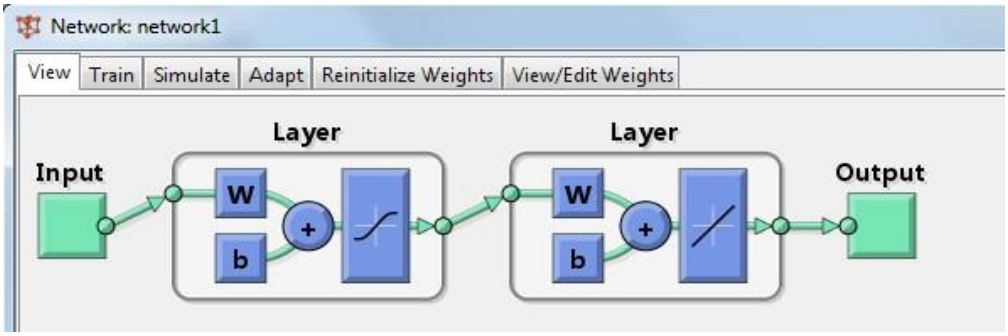


Рисунок 4 – Діалогова панель Network (структура мережі)

Навчимо мережу, при цьому в якості вхідних даних необхідно використовувати матрицю *x1x2*, а в якості цілі – вектор *T* (рис. 5).

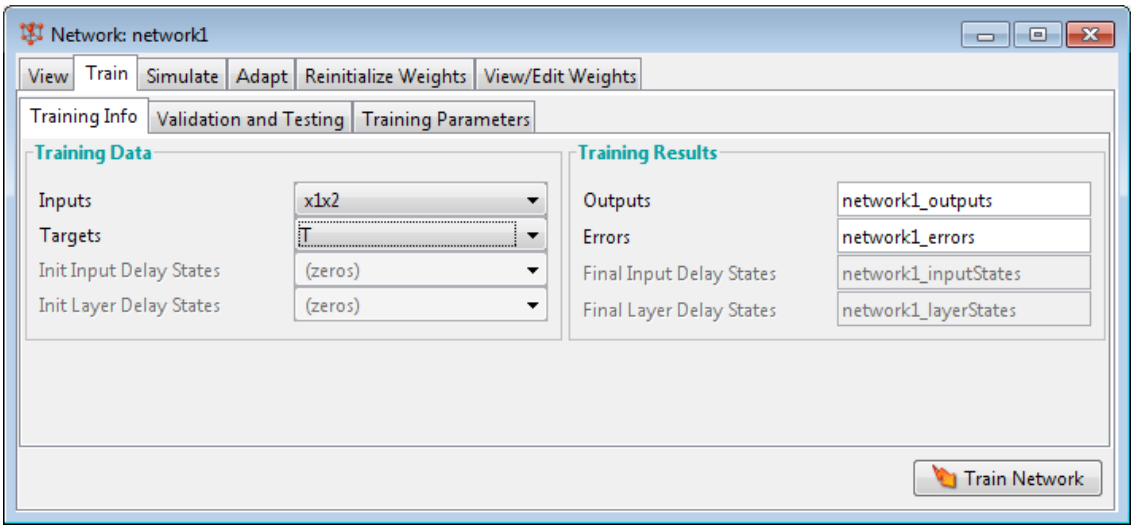


Рисунок 5 – Вікно інформації про навчальні послідовності

Для запуску навчання обираємо опцію **Train Network**. Результат навчання мережі представлений на рис. 6. Мережа навчена за 138 ітерацій до помилки 10^{-5}. Крім цього зміну функції втрат, стану, лінійної регресії між виходом і цілями мережі під час навчання можна обравши відповідні опції **Performance**, **Training State**, **Regression** в **Neural Network Training (nntraintool)** (рис. 6 - 8).

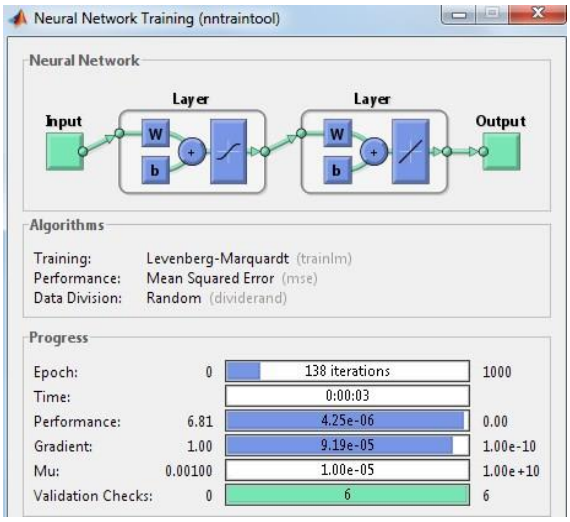


Рисунок 6 – Вікно результатів навчання нейронної мережі

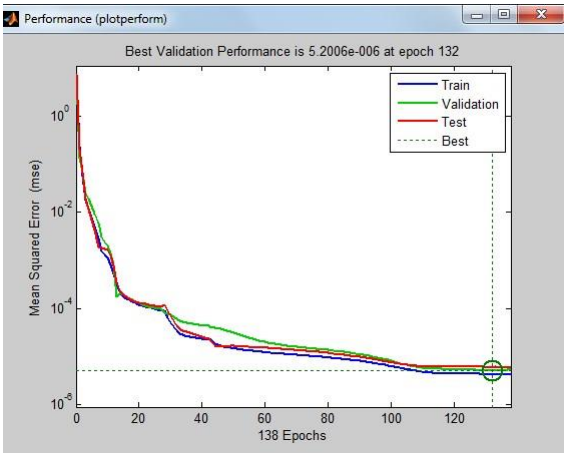


Рисунок 7 – Зміна функції втрат мережі в процесі навчання

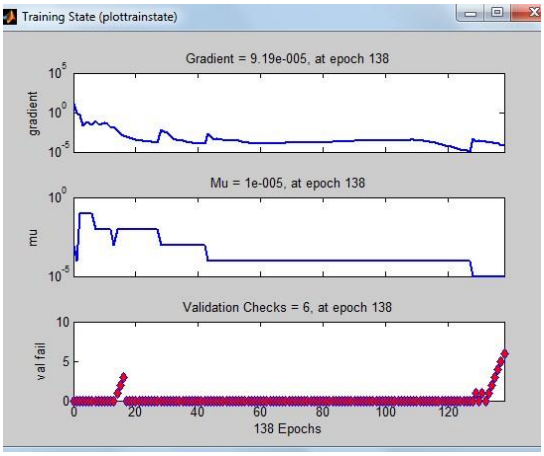


Рисунок 8 – Вікно стану навчання

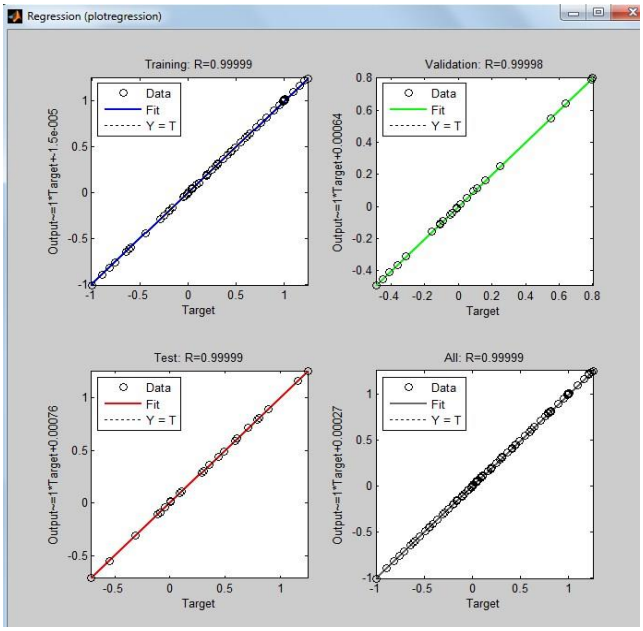


Рисунок 9. Вікно лінійної регресії між виходом ШНМ і цілями

Для оцінки якості навчання побудуємо графік помилки між бажаним значеннями на виході мережі і обчисленими. Для цього подамо на вхід мережі всю навчальну вибірку *x1x2*, скориставшись вкладкою **Simulate** і опцією **Simulate Network** (рис. 10). Результати обчислення значень виходу (симуляція) будуть записані в змінну **network1_outputs**.

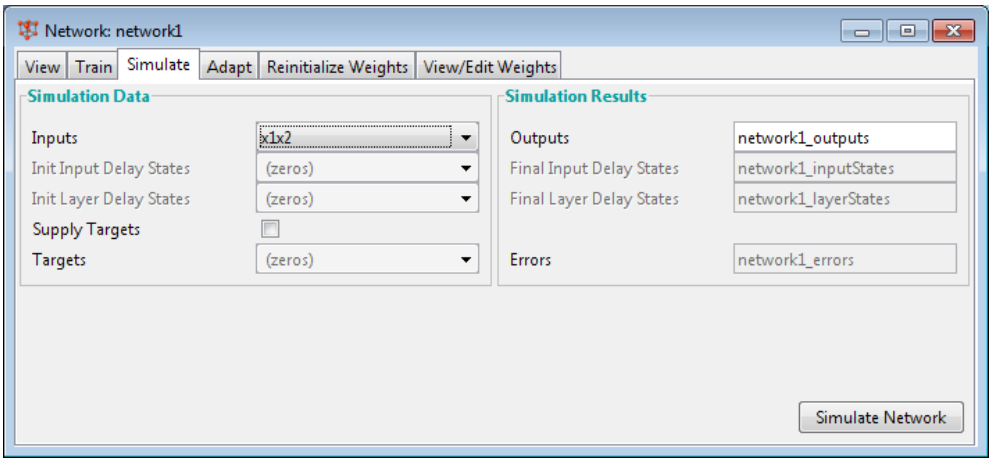


Рисунок 10 – Вікно симуляції нейронної мережі

Після цього необхідно експортувати вектор **network1_outputs** за допомогою **Network/Data Manager** в робочу область MATLAB (рис. 11).

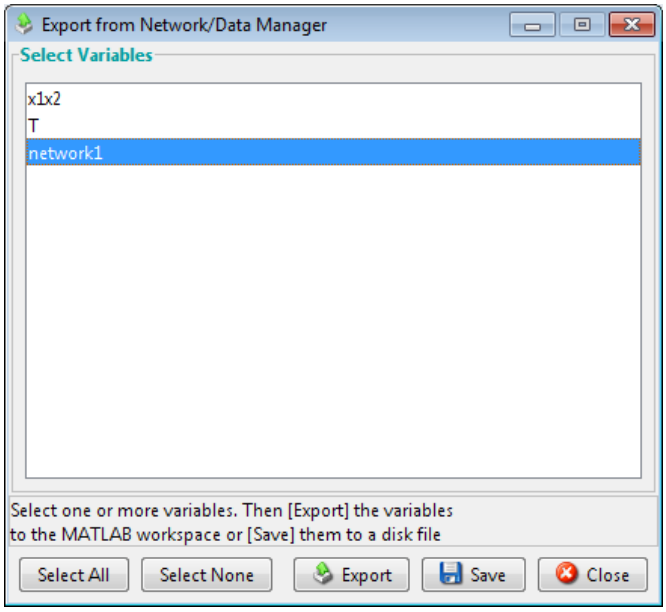


Рисунок 11 – Вікно для експорту даних у робоче середовище MATLAB або в файл

Абсолютну похибку мережі можна знайти, отримавши різницю виходу навченої мережі при симуляції network1_outputs і цілі T в робочій області:

error = network1_outputs-T;

plot(error) % Рисунок 12

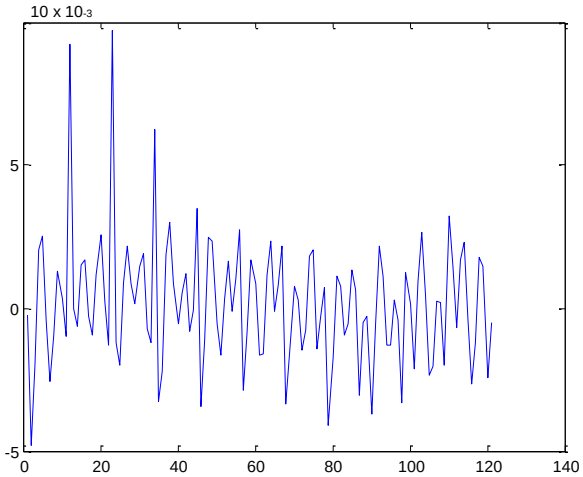


Рисунок 12 – Графік помилки нейронної мережі

2. Приклад апроксимації нелінійної функції.

Необхідно виконати апроксимацію функції наступного виду:

$$y = \sin\left(\frac{5\pi x}{N}\right) + \sin\left(\frac{7\pi x}{N}\right)$$

де x = 1 ÷ N, а N – число точок функції.

Ця крива є відрізком періодичного коливання з частотою 5π/N, модульованого за фазою гармонійним коливанням з частотою 7πN (рис. 13).

x = 1: 100;

```
y = sin (5 * pi * [1: 100] / 100 + sin (7 * pi * [1: 100] / 100));
plot (x, y) % Рисунок 13
```

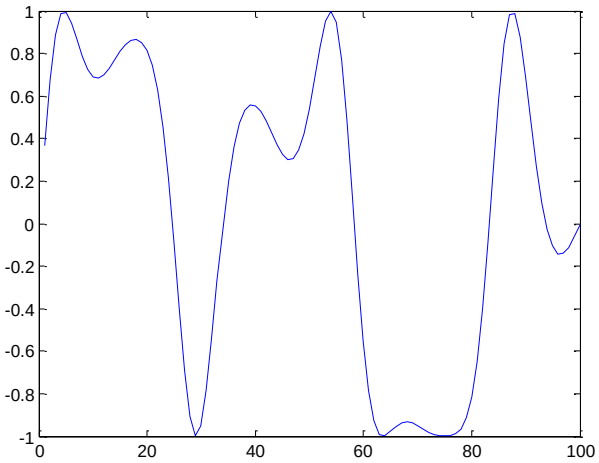


Рисунок 13 – Графік функції y = sin(5*pi*[1:100]/100+sin(7*pi*[1:100]/100))

Імпортуємо вхідні та цільові дані в nntool (рис. 14)

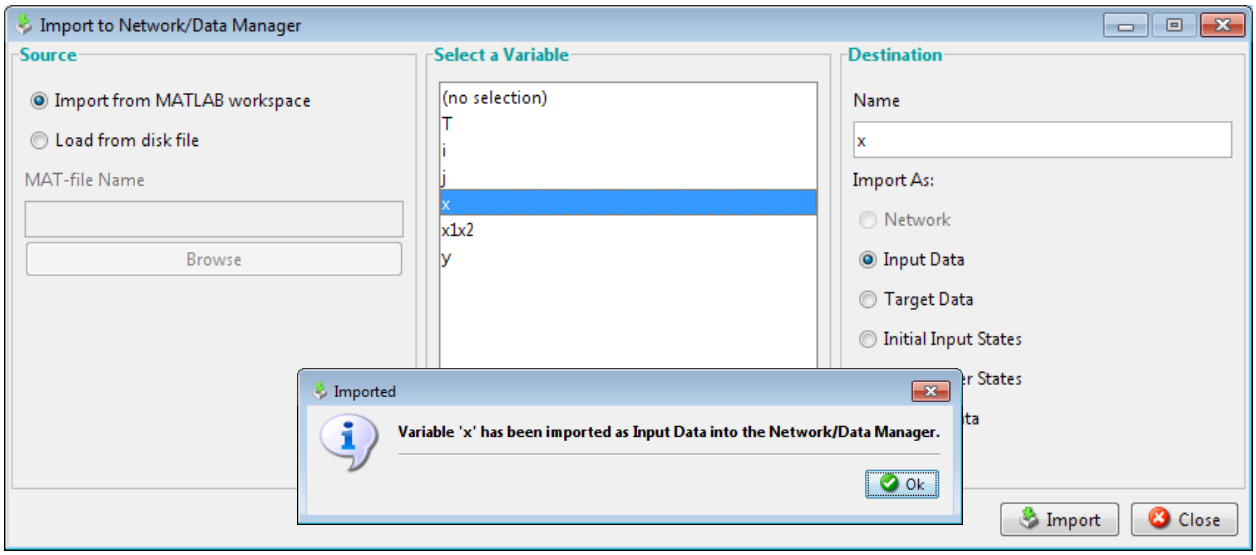


Рисунок 14. Вікно для імпорту і завантаження даних

Створимо нову мережу з ім'ям network2. Виберемо персептрон (Feed-Forward Back Propagation) с тринадцятьма сигмоїдними (TANSIG) нейронами прихованого шару і одним лінійним (PURELIN) нейроном вихідного шару. Навчання будемо проводити, використовуючи алгоритм Левенберга-Маркардта (Levenberg-Marquardt), який реалізує функція TRAINLM. Функція помилки – MSE.

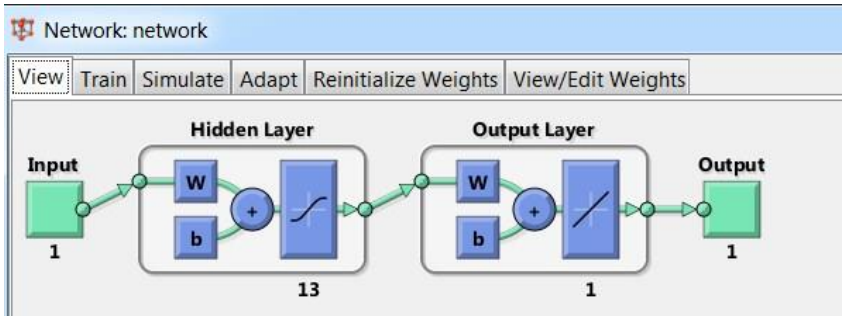


Рисунок 15 – Архітектура мережі для вирішення задачі апроксимації

Тепер можна приступити до навчання. Для цього необхідно вказати, які набори даних повинні бути використані в якості навчальних і цільових, а потім провести навчання (рис. 16-20).

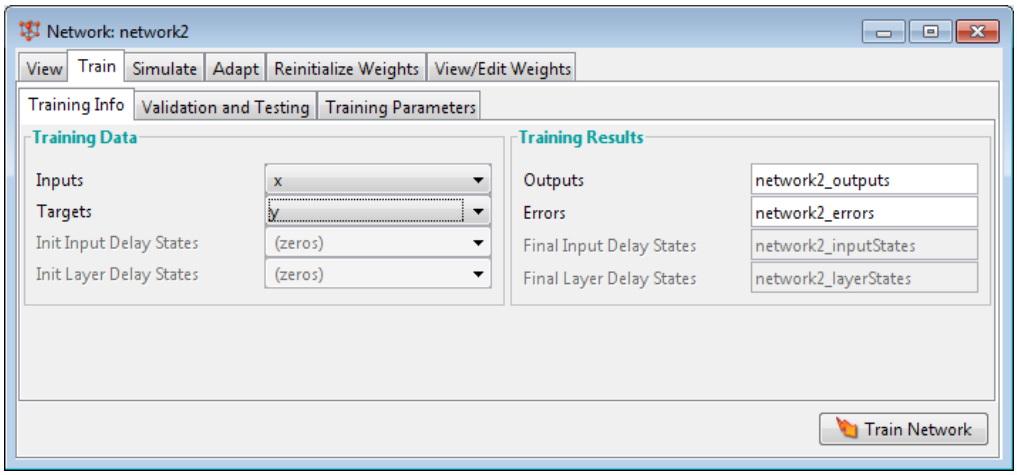


Рисунок 16 – Вікно інформації про послідовності для навчання.

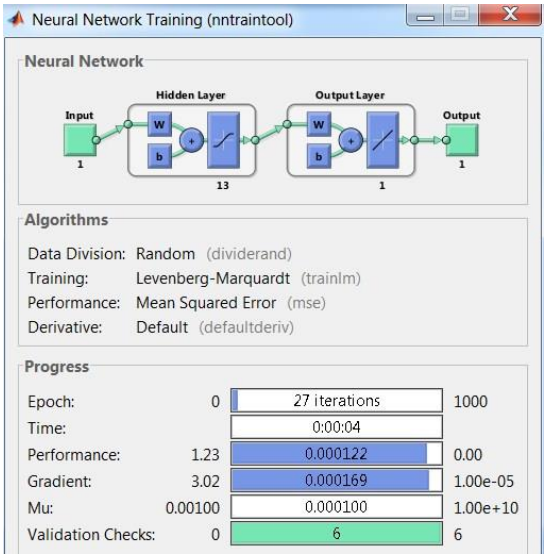


Рисунок 17 – Вікно навчання нейронної мережі.

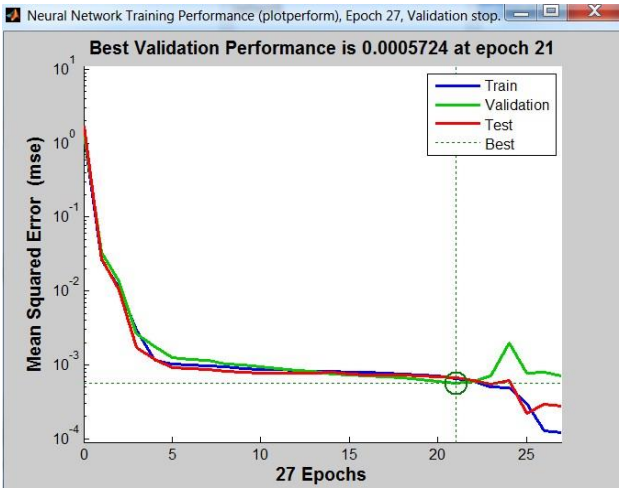


Рисунок 18 – Зміна похибки мережі в процесі навчання.

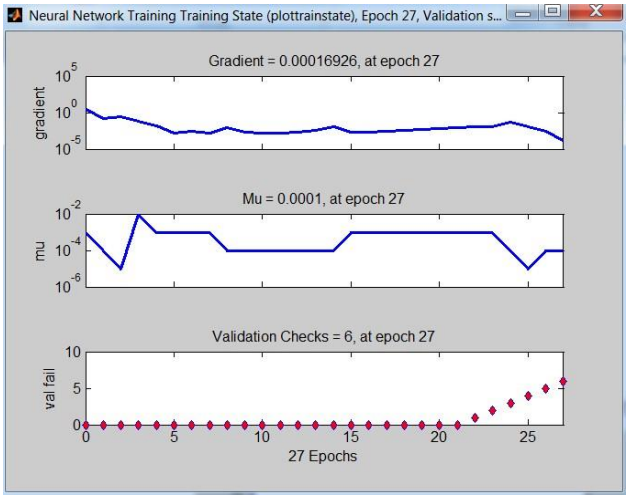


Рисунок 19 – Вікно стану навчання.

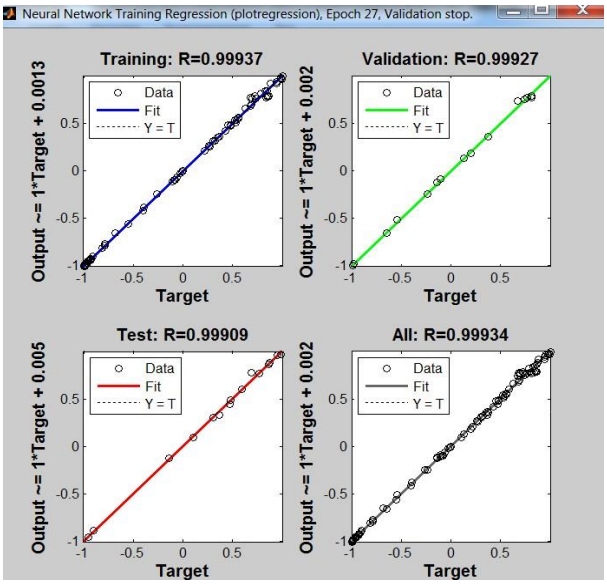


Рисунок 20 – Вікно лінійної регресії між виходом ШНМ і цілями.

Експортуємо значення виходу нейронної мережі в робочу область:

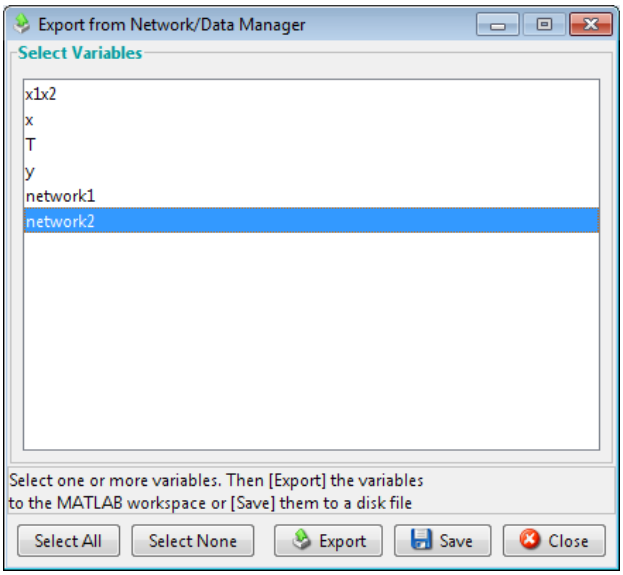


Рисунок 21 – Вікно для експорту або запису даних в файл

Побудуємо графіки функції:

y = sin(5*pi*[1:100]/100+sin(7*pi*[1:100]/100));

і виходу нейронної мережі в одній системі координат:

plot(x,y, x,network2_outputs) % Рисунок 22

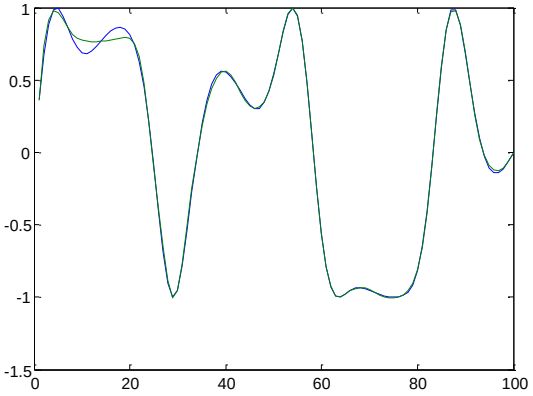


Рисунок 22 – Криві цільових даних і виходу нейронної мережі.

На рис. 23 зображений графік похибки наближення:

error = y-network2_outputs;

plot(error) % Рисунок 23

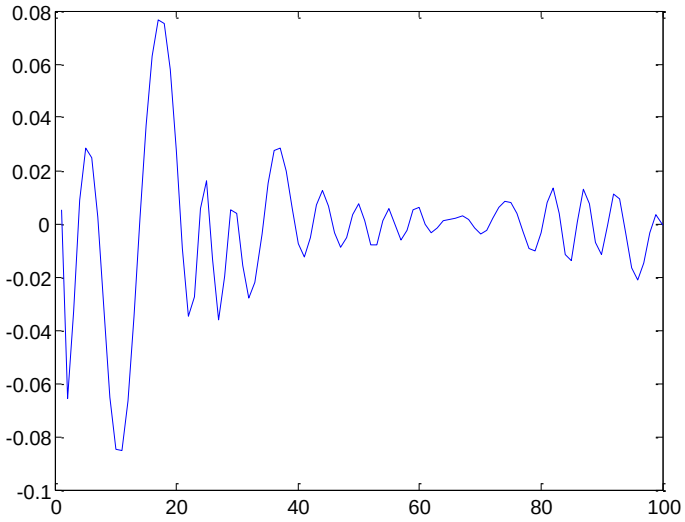


Рисунок 23 – Графік величини похибки

Точність апроксимації може бути підвищена за рахунок збільшення кількості нейронів в шарі.

3. Апроксимації синусоїдальної функції, що спотворена нормально розподіленим шумом

```
P = [-1:.05:1];
T = sin(2*pi*P)+0.1*randn(size(P));
Plot(P,T) % рисунок 24
%Створення мережі
net = newff([-1 1],[20,1],{'tansig','purelin'},'trainbr');
%Навчання мережі:
net.trainParam.epochs = 50;
net.trainParam.show = 10;
net = train(net,P,T);
gridon;
```

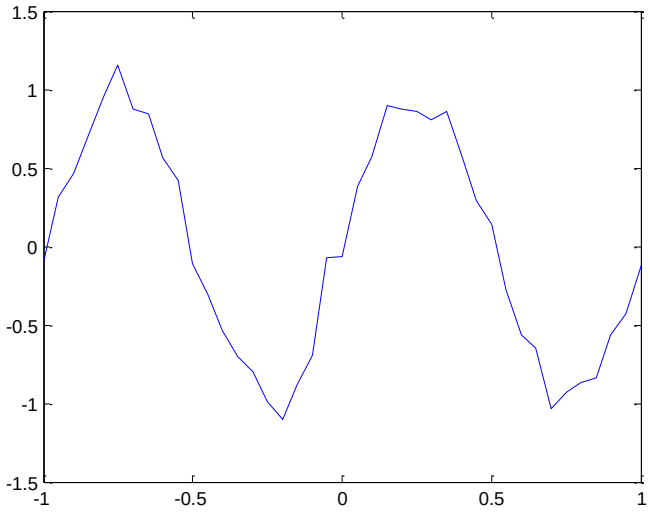


Рисунок 24 – Графік синусоїдальної функції, що спотворена нормально розподіленим шумом

Сформуємо для вирішення цього завдання двошарову нейронну мережу прямого поширення сигналу. Вхід мережі приймає значення в діапазоні від -1 до 1. Перший шар має 20 нейронів з функцією активації tansig, другий шар має один нейрон з функцією активації purelin.

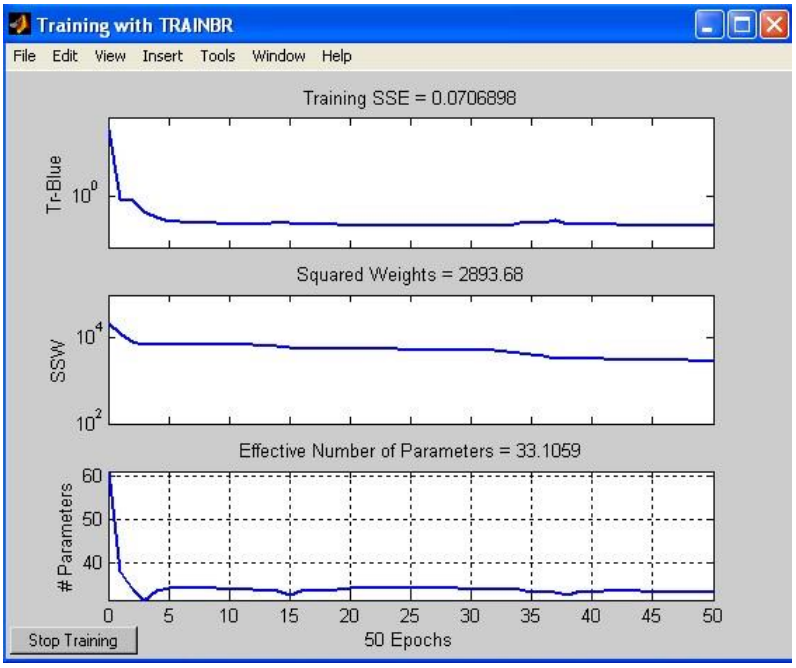


Рисунок 25 – Вікно стану навчання

Здійснимо моделювання мережі і побудуємо графіки функцій, що досліджуються

```
Y = sim(net,P);
figure(2), clf, h1 = plot(P,Y,'LineWidth',2);
hold on, set(h1,'Color',[1/2,1/2,0])
plot(P,T,'+r','MarkerSize',6,'LineWidth',2), grid on
legend('выход','вход') % рисунок26
```

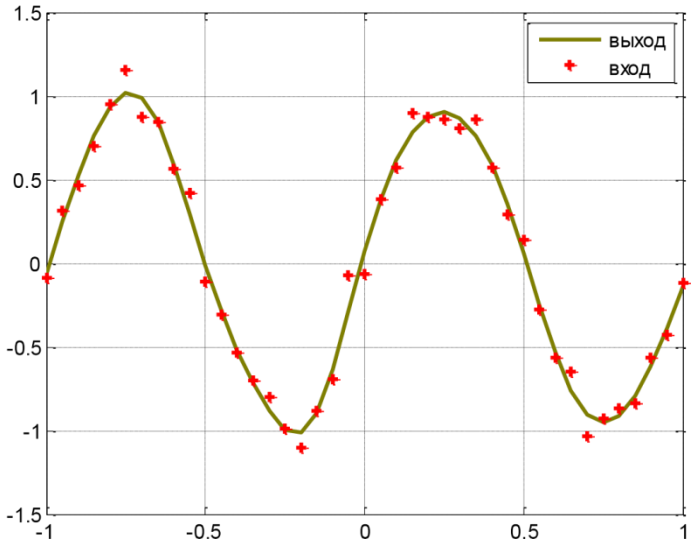


Рисунок 26 – Графіки виходу нейронної мережі і фактичних значень сигналу

Перелік рекомендованої літератури

1. Ямпольський Л.С. Нейротехнології та нейрокомп’ютерні системи: підручник / Л.С. Ямпольський, О.І. Лісовиченко, В.В. Олійник. – К.: «Дорадо-Друк», 2016. – 576 с.: іл. – Бібліогр.: с. 537-551.

2. Дьяконов В. П. MATLAB. Полный самоучитель. – М.: ДМК Пресс, 2012. – 768 с.: ил.

3. Хайкин, Саймон Нейронные сети: полный курс, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 1104 с.: ил. – Папал. Тит. Англ.